MANUEL TECHNIQUE DE PROGRAMMATION

MIDA

AFEI Sistemas y Automatización,S.A.

C/ Cartagena. 245, 4º, 1ª
08025 Barcelone (Espagne)
Tél. : (+34) 93 446 30 50
Fax : (+34) 93 446 30 51
E-mail : afei@afeisa.es
http://www.afeisa.es

Cet ouvrage a été réalisé par les techniciens ci-après, qui ont tous participé directement au développement.

Rédaction et mise en page : Jordi Vila
Santiago Losada

Supervision : Jordi Vila
Jordi Vives
Santiago Losada

Première édition (Version 00.06) December 2000

IG

IG

DP

IN

ES

FI

DF

PC

TE

Avant de nous lancer dans les explications concernant la programmation des équipements MIDA, nous nous permettons de vous souhaiter la bienvenue.

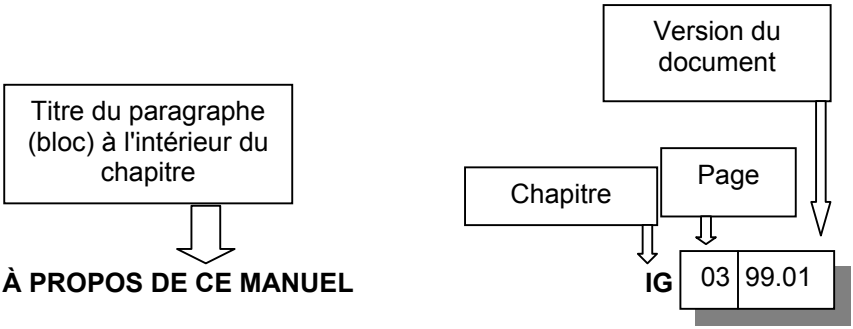
La rédaction d'un manuel de programmation est toujours une tâche difficile, car des explications qui peuvent sembler très détaillées à certains seront, au contraire, jugées trop résumées par d'autres.
Il est donc difficile de trouver un équilibre. C'est le but que nous avons tenté d'atteindre et nous espérons y être parvenus. Nous restons ouverts à toute suggestion concernant ce manuel et vous remercions par avance de votre collaboration.

La plupart des manuels recommandent la lecture intégrale de l'ouvrage et la mise en pratique des connaissances acquises grâce à des exemples, car c'est le seul moyen de disposer d'une vision globale et de maîtriser les nombreuses possibilités offertes par les instructions.

Néanmoins et comme nous le savons tous, pour des raisons liées aux délais ou au développement ou pour tout autre motif, le temps est une denrée rare, et il nous en reste peu à investir dans la lecture d'un manuel de programmation. C'est pourquoi l'équipe de AFEISA a décidé de concevoir ce manuel d'une manière différente, c'est-à-dire en regroupant les instructions selon leur utilité : FONCTIONS LOGIQUES (LD, AND, OR, etc.), TEMPORISATEURS ET COMPTEURS (TIM, CNT, etc.) INSTRUCTIONS ARITHMÉTIQUES, etc. Chaque instruction est expliquée et illustrée par un exemple que vous pouvez appliquer immédiatement, et chaque bloc d'instructions se termine par un exemple plus complet dans lequel nous avons tenté d'intégrer toutes les instructions expliquées dans le bloc en question.

Le fait que les équipements concernés puissent fonctionner comme des automates programmables ou comme des microcontrôleurs est l'une des autres difficultés que nous avons rencontrées lors de la rédaction de ce manuel. Nous avons tenté de traiter chaque instruction de manière appropriée.

Avant de poursuivre, nous pensons qu'il est nécessaire de vous fournir quelques explications concernant la structure du manuel et de chacune des pages :



REMARQUE :

La numérotation des pages reprend à chaque changement de chapitre. Chaque chapitre commence donc par la page numéro 1.

DP

Tout au long de ses 15 années d'expérience dans le domaine de la conception et de la fabrication d'équipements pour l'automatisation de processus industriels, AFEISA a su conserver l'esprit jeune, dynamique et novateur qui lui a permis de créer les équipements MIDA: un nouveau concept dans le domaine des PLC. Mais AFEISA ne s'est pas arrêtée là. Elle est allée au-delà en intégrant à ses équipements des fonctions de pesage, data-logger, de régulation industrielle, d'enregistrement de données, etc.

Les équipements MIDA impliquent l'intégration du dialogue homme-machine dans une seule unité et constituent ainsi des ensembles compacts, robustes, simples et pratiques pour l'utilisateur. Leurs différentes configurations d'entrées et de sorties, analogiques et numériques, en font des systèmes polyvalents, adaptables aux besoins précis de chaque application.

Certains équipements peuvent en outre être pourvus d'un clavier redéfinissable doté de leds d'état pour une personnalisation accrue de l'ensemble.

Toutes les unités disposent également de ports de communication série RS-232 pour la gestion de divers périphériques - imprimantes, PC de surveillance, lecteurs de codes à barres, afficheurs - ou l'établissement de connexions via modem ou radio-modem, etc.

Les ports de communication RS-485 des équipements permettent quant à eux de connecter 32 unités sur une distance de 1.200 mètres pour effectuer des contrôles partagés, centraliser les processus, superviser l'ensemble depuis un PC, etc.

AFEISA possède son propre protocole de communication (MIDABUS) et la majorité des modèles disposent du protocole standard MODBUS en sus du PROTOCOLE LIBRE qui consiste en un jeu d'instructions permettant de lire et d'écrire d'autres protocoles.
Les équipements sont donc parfaitement transparents pour le PC, d'où une disponibilité totale pour l'acquisition et la gestion des données, sans nécessité d'interrompre l'exécution du programme.

Ils disposent en outre d'un puissant jeu d'instructions logiques et arithmétiques, de registres entiers (16 bits) et à virgule flottante (32 bits), et d'instructions de commande des périphériques d'affichage et du clavier, selon les prestations de chaque équipement.

La programmation des unités MIDA est extrêmement simple et son langage de haut niveau facilite la réalisation de projets complexes car il est possible d'exécuter des programmes contenant des sous-routines, des sauts conditionnels ou inconditionnels et des registres indexés, et de structurer le programme à l'aide d'une procédure principale et de routines secondaires.

Vous trouverez, ci-après, diverses définitions qui vous aideront à comprendre les explications fournies dans ce Manuel technique.

Programme MIDA	Programme de type PLC (automate programmable) résident de la mémoire FLASH EPROM de l'équipement et créé par l'intégrateur.
Mémoire-tampon intermédiaire	Zone de mémoire RAM de l'équipement dans laquelle s'effectue la préparation des envois de données aux périphériques d'affichage ou aux ports de communication (imprimantes, PC, etc.).
Pile logique	Zone de mémoire RAM de l'équipement dans laquelle les états de 1 bit (relais, entrées/sorties numériques) sont provisoirement stockés pour être gérés par le programme MIDA.
Pile arithmétique	Zone de mémoire RAM de l'équipement dans laquelle les états de 16 bits (données entières) ou de 32 bits (données à virgule flottante) sont provisoirement stockés pour être gérés par le programme MIDA.
Pile de sous-routines	Zone de mémoire RAM de l'équipement dans laquelle les adresses de retour des sous-routines sont provisoirement stockées.
Pseudo-instruction	Instruction du programme MIDA affectant le travail du compilateur.
Compilateur	Programme dont la fonction consiste à traduire le programme MIDA écrit par l'utilisateur en code objet exécutable par l'équipement.
Instruction	Principale composante d'une ligne de programme MIDA. Elle indique l'action à réaliser sur les opérandes de cette ligne de programme.

Mnémonique	Nom de chacune des instructions de programmation.
Opérande	Partie de la ligne de programme à laquelle s'applique l'instruction. L'opérande représente un relais, un registre, une entrée, une sortie, etc.
Jeu d'instructions	Mode de programmation basé sur l'utilisation des mnémoniques et des opérandes en tant qu'éléments d'identification des instructions et consistant en l'édition d'une série d'instructions par ordre d'exécution.
Schéma à relais ou "Ladder"	Mode de programmation basé sur l'utilisation de symboles graphiques qui simulent les schémas électriques.
Programme interprète	C'est le programme maître du système qui permet de traduire le programme MIDA créé par l'intégrateur en instructions exécutées par le microprocesseur.

Les équipements MIDA disposent de plusieurs types de mémoires :

✓ **EPROM-FLASH**

Mémoire où sont stockés le programme interprète et celui, de type PLC, créé par l'utilisateur. Ces deux programmes peuvent être transmis depuis un PC via un port série. Il est donc possible d'actualiser le programme interprète et le programme MIDA sans remplacer l'EPROM.

✓ **NOVRAM**

Mémoire Non Volatile où sont stockées les données intermédiaires liées à l'exécution du programme MIDA (relais, registres entiers et à virgule flottante). Ce type de mémoire ne dépendant pas de la batterie au Ni-Cd, elle peut être utilisée pour stocker des données importantes pour l'exécution du programme (codes d'accès, constantes de conversion, etc.).

La NOVRAM possède non seulement les avantages de l'EEPROM, mais aussi un atout supplémentaire : le nombre d'écritures réalisables n'est pas limité. Elle est partagée en deux zones : une pour les registres entiers de 16 bits avec signe et une pour les registres de 32 bits à virgule flottante.

✓ **RAM**

Mémoire où sont stockées les données intermédiaires liées à l'exécution du programme MIDA (relais, registres). De par sa construction, ce type de mémoire est non volatile. L'équipement dispose néanmoins d'une batterie au Ni-Cd pour sauvegarder les données qui sont stockées dans cette mémoire. Cette batterie possède une autonomie de trois mois lorsque l'équipement est hors tension.

La mémoire RAM est, elle aussi, constituée de deux zones :

- **RAM utilisateur.**
- **RAM temporaire.**

- RAM UTILISATEUR
La RAM utilisateur est une mémoire non volatile. Elle est protégée par la batterie au Ni-Cd.
Elle est partagée en plusieurs zones :

- *Zone de stockage réservée au programme interprète de l'équipement.*
- *Zone de stockage des positions à 1 bit (relais).*
- *Zone de stockage des positions à 16 bits (registres entiers).*
- *Zone de stockage des positions à 32 bits (registres à virgule flottante).*

- RAM TEMPORAIRE
Les équipements MIDA disposent de 3 ***piles (stacks)*** indépendantes et d'une mémoire-tampon intermédiaire :

- **PILE LOGIQUE (1 bit):** pour l'échange d'états intermédiaires de 1 bit (relais, entrées, sorties numériques). Elle possède une capacité de 50 niveaux.

- **PILE ARITHMÉTIQUE (16 bits):** pour l'échange de données de 16 et 32 bits (respectivement entiers et à virgule flottante). Elle possède une capacité de 50 niveaux.

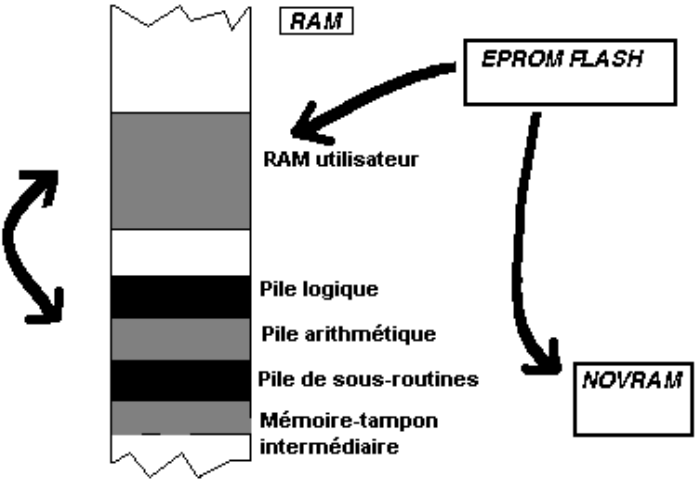
- **PILE DE SOUS-ROUTINES (16 bits):** pour le stockage des adresses de retour des sous-routines (instructions "CALL"). Elle possède une capacité de 50 niveaux.

Les piles sont de type LIFO (dernier arrivé - premier sorti). Nous considérons donc, dans les explications, que le dernier état/la dernière donnée occupe la position supérieure de la pile et est donc le premier/la première à sortir.

Les équipements MIDA disposent également d'une ***MÉMOIRE-TAMPON INTERMÉDIAIRE*** (zone de stockage des données temporaires) où s'effectue la préparation de l'affichage, ainsi que celle

des transmissions et des réceptions via port série. Elle possède ***une longueur de 132 octets.***

Toutes les zones de mémoire décrites sont protégées contre le débordement.



Lorsque le programme exécute l'instruction END, tous les niveaux des piles logiques, arithmétique et de sous-routines sont vérifiés pour détecter d'éventuelles erreurs de programmation. Le cas échéant, elles sont réinitialisées.

Dans le cas de la mémoire-tampon intermédiaire, le programme interprète empêche lui-même toute écriture en dehors des limites de cette mémoire.

Les modes dans lesquels les équipements MIDA peuvent se trouver sont de deux types :

✓ MODE TRAVAIL

Il s'agit du **mode normal de fonctionnement par défaut** de l'équipement, c'est-à-dire du mode depuis lequel le programme est exécuté. Il est également connu sous le nom de mode RUN.

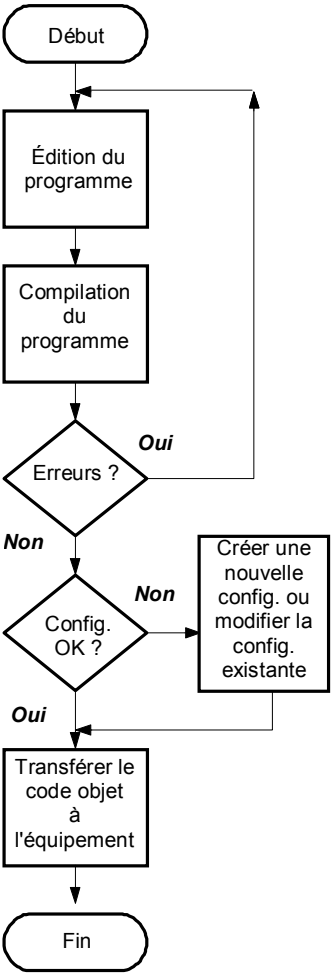
L'équipement passe dans ce mode de travail ***lors de sa mise sous tension*** et lorsque, l'équipement étant en MODE ARRÊT (STOP), l'utilisateur lance ***une réinitialisation de programme*** (reset) par l'intermédiaire du clavier ou d'un port de communication.

✓ MODE ARRÊT

Lorsque l'équipement se trouve en mode ARRÊT, l'utilisateur peut interrompre l'exécution du programme par l'intermédiaire du clavier ou d'un port de communication. C'est le mode approprié pour effectuer des tests et des vérifications relatives au programme.

Lorsqu'il se trouve dans ce mode, l'équipement exécute le programme normalement, mais suit les instructions d'interruption et d'accès aux fonctions internes fournies par l'intermédiaire du clavier : test de l'équipement et affichage interne des registres.

La programmation de l'équipement s'effectue grâce à un jeu d'instructions.



Pour programmer les équipements MIDA, vous devez tout d'abord créer le code de programme sur un éditeur ASCII standard, le compiler pour obtenir le code objet et transférer celui-ci à l'équipement via un port série.

Vous pouvez également utiliser un éditeur nommé MIDAedit qui fonctionne sous MIDAtools. Introduisez le programme qui doit être exécuté par l'équipement MIDA dans cet éditeur, compilez-le, puis transférez-le à l'équipement via un port série.
Signalons aussi l'existence d'un puissant outil d'aide et de support au programmeur, appelé MIDAvisual.

MIDAvisual est un logiciel d'affichage et de surveillance qui permet de lire et de modifier n'importe quel registre interne de l'équipement MIDA, de détecter les erreurs internes et d'éditer le programme. C'est un outil idéal pour l'élimination et la détection d'erreurs dans les programmes développés à l'aide des unités industrielles de commande MIDA.

Les détails de la programmation sous MIDAtools à l'aide de MIDAedit ou de MIDAvisual sont décrits dans les manuels fournis avec chacun de ces outils.

✓ Édition de programmes sous DOS

Il est important d'écrire les programmes à l'aide d'un éditeur ASCII standard pour PC n'introduisant pas de caractères de contrôle (n'employez pas d'éditeurs de type Word, WordPerfect, WordPad, etc.).
Le programme étant écrit, enregistrez-le avec l'extension .PRG.

Le nombre de lignes d'un programme peut être limité. Il varie en fonction du modèle d'équipement MIDA dont vous disposez (consultez le Manuel Utilisateur correspondant).

Une ligne de programme peut comporter au maximum : un mnémonique, trois étiquettes (opérandes) et un commentaire :

- **Mnémonique :** instruction devant être exécutée par l'équipement.
- **Étiquette :** nom symbolique définissant un relais, un registre, une constante, un texte ou un saut de ligne.
- **Opérande :** indique sur quel(le) relais, registre, constante ou saut de ligne le mnémonique doit être exécuté.
- **Commentaire :** le caractère ***;*** indique au compilateur que ce qui suit est un simple commentaire destiné au programmeur. Les commentaires peuvent être insérés à n'importe quel endroit d'une ligne de programme.

Pour de plus amples détails concernant le fonctionnement du compilateur et l'émetteur de programmes, consultez le document joint à la disquette de programmation.

✓ Pseudo-instructions de compilation

Certaines instructions (mnémoniques) n'ont d'effet que sur le compilateur. Il s'agit des pseudo-instructions de compilation :

EQU	**LITE**	**MIDA**
INTER	**END_INTER**	**FILE**

Les pseudo-instructions de compilation **INTER** et **END_INTER** sont décrites en détail dans le chapitre "Interruptions Software" et la pseudo-instruction **FILE** dans le chapitre "Base de données" de ce manuel.

- **EQU :** Permet d'assigner des noms symboliques (étiquettes) aux relais, aux registres, aux constantes et aux opérandes. Une fois définie, l'étiquette peut être utilisée à la place de l'expression équivalente. Exemple : supposons que l'entrée numérique 8 soit assignée à une entrée d'activation et que la sortie numérique 100 soit assignée à une pompe. L'expression correspondante sera la suivante :

```
activation   equ   8
pompe        equ   100

LD           activation
OUT          pompe
```

Important :

Prenez soin de ne laisser aucun espace libre entre la marge gauche et le premier caractère de l'étiquette lorsque vous éditez celle-ci. Vous devez également laisser un espace libre entre l'étiquette et le mnémonique, entre le mnémonique et l'opérande et entre les différents opérandes lorsque vous éditez le programme.

- **LITE :** Permet d'assigner à des libellés des noms symboliques (étiquettes) au lieu de numéros d'ordre. Exemple : supposons que le libellé à afficher est CONTRÔLE A, son expression peut être la suivante :

```
contr_a      lite    "CONTRÔLE A"
CLEAR
DISL         contr_a
COM          0
```

Le libellé doit être placé entre guillemets et sa longueur dépend de l'équipement MIDA utilisé. Si vous souhaitez laisser le libellé en blanc, laissez un espace libre entre les guillemets.

Les libellés doivent être édités consécutivement dans la table de définition des étiquettes.

Pour accéder à un libellé à l'intérieur d'un programme, vous pouvez l'appeler soit par son numéro d'ordre, soit par l'étiquette qui lui est assignée dans la table.

Le premier libellé de la table porte le numéro d'ordre 0. Le numéro d'ordre le plus élevé dépend quant à lui de l'équipement que vous êtes en train de programmer.

Important :

Prenez soin de ne laisser aucun espace libre entre la marge gauche et le premier caractère de l'étiquette lorsque vous éditez celle-ci. Vous devez également laisser un espace libre entre l'étiquette et le mnémonique, entre le mnémonique et l'opérande et entre les différents opérandes lorsque vous éditez le programme.

- **MIDA :** Permet de spécifier au compilateur le modèle de MIDA auquel le programme est destiné pour qu'il puisse le compiler et le préparer de manière appropriée. Le programme peut ensuite être transmis à l'équipement en question. Exemple : supposons que le programme est destiné à un équipement MIDA 14 :

```
        MIDA 14

activation  equ  8
pompe       equ  100

LD          activation
OUT         pompe
```

Important :

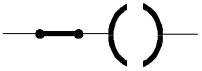
Prenez soin de laisser un espace libre entre la marge gauche et le premier caractère de cette pseudo-instruction, ainsi qu'entre celle-ci et le chiffre correspondant au modèle de MIDA (variable selon l'équipement).

Les symboles utilisés dans ce manuel sont les suivants :

LECTURE DU CONTACT D'UN RELAIS (LD)

LECTURE DU CONTACT D'UN RELAIS COMPLÉMENTÉ (LDNT)

DÉCONNEXION D'UN RELAIS (RESET)

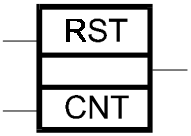


CONNEXION D'UN RELAIS (SET)

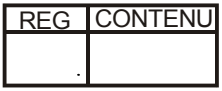
SORTIE NUMÉRIQUE OU RELAIS INTERNE (OUT)

SORTIE NUMÉRIQUE COMPLÉMENTÉE OU RELAIS INTERNE COMPLÉMENTÉ (OUTNT)

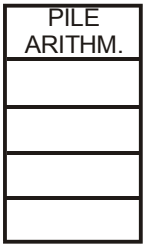
TEMPORISATEUR (TIM)



COMPTEUR (CNT)



REGISTRE ET SON CONTENU



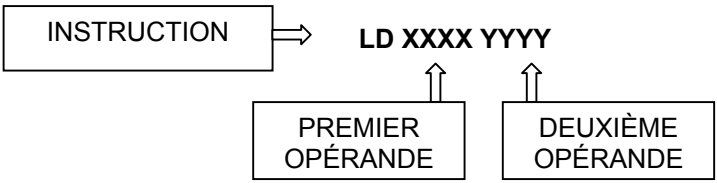
PILE ARITHMÉTIQUE

IN

Avant de commencer à détailler chacune des instructions de programmation, nous aimerions vous fournir quelques explications qui, nous l'espérons, contribueront à une meilleure compréhension de ce manuel.

IMPORTANT :
Ce manuel décrit toutes les instructions utilisées par les différents équipements MIDA, mais cela ne signifie pas que toutes ces instructions sont disponibles pour chaque modèle MIDA.
Nous vous conseillons donc de vérifier si les exemples, les instructions et l'adressage des relais/registres de mémoire décrits dans ce manuel sont disponibles sur l'équipement MIDA que vous vous disposez à programmer.
Pour déterminer les instructions, les relais/registres et les limites disponibles sur chaque modèle de MIDA, consultez le Manuel de l'utilisateur correspondant.

Les instructions peuvent contenir au maximum deux opérandes qui se présentent de la manière suivante :



Pour vous aider à localiser et à identifier les informations, nous avons créé une fiche type portant la description détaillée de chacune des instructions utilisées par les équipements MIDA. Vous trouverez, ci-après, un schéma de la structure de cette fiche :

DESCRIPTION DE LA FICHE D'INSTRUCTION **IN-DI** 2 99.01

LD ⇐ Nom de l'instruction

LD XXXX

Mnémonique

MNÉMONIQUE : **LD**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie.**
CODE INSTRUCTION : **01**

Éléments descriptifs de l'instruction

Description et effet de l'instruction

DESCRIPTION :

CHARGE, dans la pile logique, l'état du relais désigné par l'opérande.
Incrémente la pile logique de 1 niveau (+1).

Exemple pratique

☑ Exemple :

Commentaires

```
LD    1     ;Charge l'état de l'entrée digital 1 dans la pile logique.
OUT   101   ;Décharge le dernier état de la pile logique (celui de
            ;l'entrée 1) sur la sortie numérique 101.
```

Représentation graphique de l'exemple décrit

Les instructions utilisées par les équipements MIDA peuvent être classées dans les groupes détaillés ci-après :

INSTRUCTIONS LOGIQUES ET DE SAUT

Instructions permettant d'assurer le contrôle logique des automatismes d'une machine ou d'un système. Elles sont associées à des registres de 1 bit.

LD	ANDNT	OUT	LDX
LDNT	ORNT	OUTNT	OUTX
AND	ANDLD	SET	XOR
OR	ORLD	RESET	JZ
JNZ			

INSTRUCTIONS DE TEMPORISATION ET DE COMPTAGE

Instructions de temporisation et de comptage, associées à des registres entiers de 16 bits.

TIM	CNT	TIMR	CNTR

INSTRUCTIONS DE TRAITEMENT ARITHMÉTIQUE DE 16 BITS

Instructions permettant de réaliser des opérations arithmétiques et associées à des registres entiers (16 bits avec signe).

MOVCI	MOVIX	ADDI	ADDC	INC
MOVRI	STOIX	SUBI	SUBC	SETRI
STOI		MULI	MULC	STOFI
		DIVI	DIVC	

INSTRUCTIONS DE TRAITEMENT ARITHMÉTIQUE DE 32 BITS

Instructions permettant de réaliser des opérations arithmétiques et associées à des registres à virgule flottante (32 bits selon le format IEEE).

MOVCF	MOVFX	ADDF
MOVRF	STOFX	SUBF
STOF	MOVIF	MULF
		DIVF

INSTRUCTIONS DE COMPARAISON ARITHMÉTIQUE

Instructions permettant de réaliser des comparaisons entre registres entiers (16 bits) ou à virgule flottante (32 bits).

CPEF	CPGF	CPEI	CPGI
CPGEF	CPLF	CPGEI	CPLI
CPLEF		CPLEI	

INSTRUCTIONS D'AFFICHAGE ET D'IMPRESSION

Instructions permettant d'afficher des données alphanumériques, de les imprimer, de les transmettre via un port série, etc.

CLEAR	DISRI	DISIX	DISL
LOC	DISFX	DISRF	DISLX
LOCX	DISCX	DISCH	COM

INSTRUCTIONS DE CLAVIER

Instructions permettant de détecter l'activation des touches ou d'introduire des données alphanumériques dans les registres internes de l'équipement.

INK	INI	INF	INICF
	INPIX	INPFX	INPCX

INSTRUCTIONS D'HORLOGE

Instructions permettant d'afficher et de modifier l'horloge interne de l'équipement.

TIME	DATE	CLOCK	CLKP

INSTRUCTIONS DE COMMANDE DE FLUX DE PROGRAMME

Instructions de saut et de sous-routines qui permettent d'élaborer des codes de programme structurés dans le cadre d'une procédure principale et de procédures secondaires.

CALL	JMP	END	NOP
RET			

INSTRUCTIONS DE DÉTECTION DE FLANCS

Instructions permettant de détecter les flancs.

FLANC			

INSTRUCTIONS DE TRAITEMENT DE FICHIERS

Instructions permettant de manipuler les fichiers stockés dans la zone de mémoire réservée à cette fin.

WRITE	READ		

INSTRUCTIONS LIÉES AUX FONCTIONS INTERNES DE L'ÉQUIPEMENT

Instructions permettant d'appeler une fonction interne de l'équipement.

FUNC			

PSEUDO-INSTRUCTIONS DE COMPILATION

Instructions fournissant au compilateur des indications concernant les opérations spéciales pour l'obtention du code objet exécutable correspondant.
Après compilation du programme, ces instructions n'apparaissent plus dans le code objet et n'ont aucun effet pendant son exécution.

EQU	LITE	MIDA	FILE
INTER	END_INTER		

INSTRUCTIONS D'INTERRUPTIONS SOFTWARE

Instructions permettant de contrôler et de gérer les interruptions logicielles.

INTER	END_INTER	IRET	

INSTRUCTIONS DE PROTOCOLE DE COMMUNICATION

Instructions permettant de manipuler les données du port de communication série par l'intermédiaire du Protocole libre.

DISB	LEBC	COM	

Les fonctions logiques sont un ensemble d'instructions qui permettent de programmer des registres de 1 bit (relais internes, entrées et sorties numériques).

Pour programmer ces instructions, vous pouvez utiliser n'importe quelle méthode facilitant la programmation de dispositifs logiques : schémas à relais, à contacts et à bobines (LADDER), algèbre booléenne, tables de Karnough, Graffcet, etc.

Vous pouvez éditer le code du programme de différentes manières :

- Programmation avec MIDAedit (environnement MIDATools pour Windows), qui permet de programmer à l'aide d'un schéma à relais (Ladder) et d'un jeu d'instructions ou en combinant les deux.

- Programmation à l'aide d'un éditeur ASCII standard et d'outils de travail sous DOS.

Nous décrivons, ci-après, chacune des instructions de ce groupe :

LD	**ANDNT**	**OUT**	**LDX**
LDNT	**ORNT**	**OUTNT**	**OUTX**
AND	**ANDLD**	**SET**	**XOR**
OR	**ORLD**	**RESET**	**JZ**
JNZ			

LD

LD XXXX

MNÉMONIQUE : **LD**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **01**

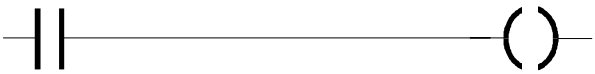
DESCRIPTION :

CHARGE, dans la pile logique, l'état du relais désigné par l'opérande.
Incrémente la pile logique de 1 niveau (+1).

☑ Exemple :

```
LD    1     ;Charge l'état de l'entrée numérique 1 dans la pile
            ;logique.
OUT   101   ;Décharge le dernier état de la pile logique (celui de
            ;l'entrée 1) sur la sortie numérique 101.
```

RELAIS 1 **RELAIS 101**



LDNT

LDNT XXXX

MNÉMONIQUE : **LDNT**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **02**

DESCRIPTION :

CHARGE, dans la pile logique, l'état complémenté du relais désigné par l'opérande.
Incrémente la pile logique de 1 niveau (+1).

☑ Exemple :

```
LDNT 1     ;Charge l'état complémenté de l'entrée numérique 1 dans
           ;la pile logique.
OUT  102   ;Décharge le dernier état de la pile logique (état
           ;complémenté de l'entrée 1) sur la sortie numérique 102.
```

RELAIS 1 **RELAIS 102**

LDX

LDX XXXX

MNÉMONIQUE : **LDX**
OPÉRANDE XXXX : **Registre entier**
CODE INSTRUCTION : **15**

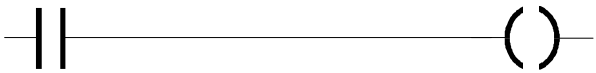
DESCRIPTION :

CHARGE, dans la pile logique, l'état du relais désigné par l'opérande.
L'opérande est le registre entier qui contient l'adresse du relais à charger.
Incrémente la pile logique de 1 niveau (+1).

☑ Exemple :

```
MOVCI   5     ;Charge la donnée 5 dans la pile.
STOI    300   ;Charge la donnée 5 dans le registre 300.
LDX     300   ;Charge l'état du relais désigné dans le registre
              ;300, dans le cas présent, l'entrée 5.
OUT     103   ;Décharge le dernier état de la pile logique (celui
              ;de l'entrée 5) sur la sortie numérique 03.
```

RELAIS 5 **RELAIS 103**



XOR

XOR XXXX

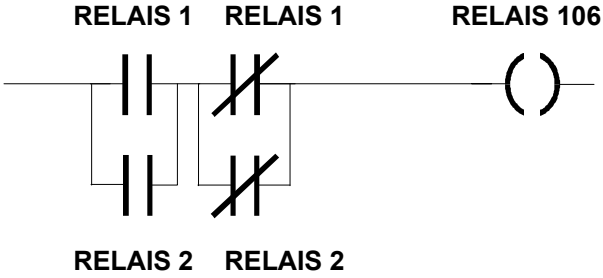
MNÉMONIQUE : **XOR**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **20**

DESCRIPTION :

OR EXCLUSIF entre le dernier état de la pile logique et l'opérande.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
LD   1     ;Charge l'état de l'entrée numérique 1 dans la pile
           ;logique.
XOR  2     ;OR EXCLUSIF entre les entrées 1 et 2.
OUT  106   ;L'état de l'entrée 1 passe sur la sortie numérique 106.
```



AND

AND XXXX

MNÉMONIQUE : **AND**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **03**

DESCRIPTION :

Effectue un AND entre le dernier état de la pile logique et l'opérande, et place le résultat dans la pile logique.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
LD   1     ;Charge l'état de l'entrée numérique 1 dans la pile
           ;logique.
AND  2     ;AND logique entre les entrées 1 et 2.
OUT  302   ;Le résultat passe sur le relais interne 302.
```

RELAIS 1 **RELAIS 2** **RELAIS 302**

OR

OR XXXX

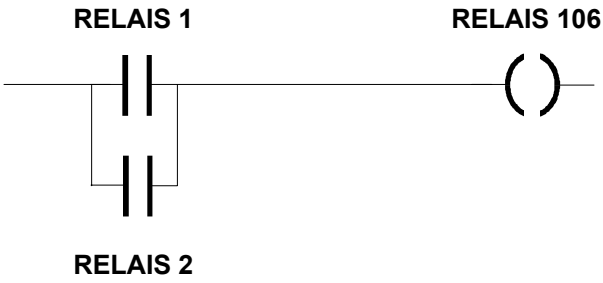
MNÉMONIQUE : **OR**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **04**

DESCRIPTION :

Effectue un OR entre le dernier état de la pile logique et l'opérande, et place le résultat dans la pile logique.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
LD   1     ;Charge l'état de l'entrée numérique 1 dans la pile
           ;logique.
OR   2     ;OR logique entre les entrées 1 et 2.
OUT  106   ;Le résultat du OR passe sur la sortie numérique 106.
```



ANDNT

ANDNT XXXX

MNÉMONIQUE : **ANDNT**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **05**

DESCRIPTION :

Effectue un AND entre le dernier état de la pile logique et l'opérande complémenté, et place le résultat dans la pile logique.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
LD      1     ;Charge l'état de l'entrée numérique 1 dans la pile
              ;logique.
ANDNT   2     ;AND entre l'entrée 1 et l'entrée 2 complémentée.
OUT     104   ;Le résultat du AND passe sur la sortie numérique
              ;104.
```

RELAIS 1 **RELAIS 2** **RELAIS 104**

ORNT

ORNT XXXX

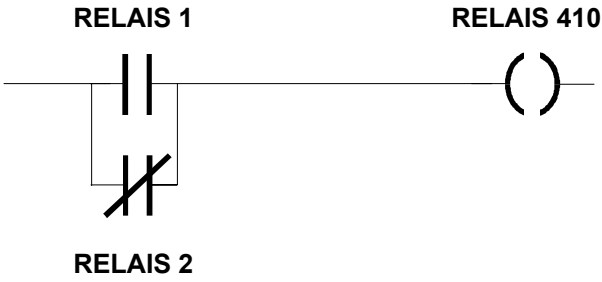
MNÉMONIQUE : **ORNT**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **06**

DESCRIPTION :

Effectue un OR entre le dernier état de la pile logique et l'opérande, et place le résultat dans la pile logique.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
LD   1     ;Charge l'état de l'entrée numérique 1 dans la pile
           ;logique.
ORNT2      ;OR logique entre l'entrée 1 et l'entrée 2 complémentée.
OUT  410   ;Le résultat du OR passe sur le relais interne 410.
```



ANDLD

MNÉMONIQUE : **ANDLD**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **07**

DESCRIPTION :

Effectue un AND logique entre les deux derniers états de la pile logique et place le résultat dans celle-ci.
Décrémente la pile logique de 1 niveau (-1).

☑ Exemple :

```
LD    1     ;Charge l'état de l'entrée numérique 1 dans la pile
            ;logique.
LD    250   ;Charge l'état du relais interne 250 dans la pile logique.
ANDLD       ;Ferme le bloc logique en appliquant un AND entre
            ;l'entrée 1 et le relais interne 250.
OUT   104   ;Le résultat du AND passe sur la sortie numérique 104.
```

RELAIS 1 **RELAIS 250** **RELAIS 104**

ORLD

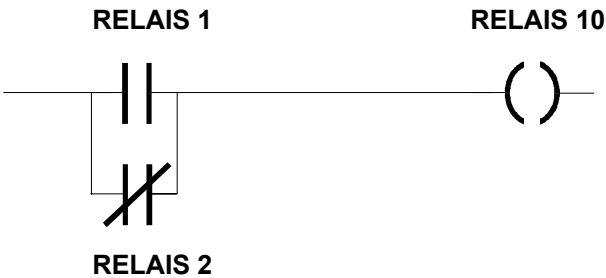
MNÉMONIQUE : **ORLD**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **08**

DESCRIPTION :

Effectue un OR logique entre les deux derniers états de la pile logique et place le résultat dans celle-ci.
Décrémente la pile logique de 1 niveau (-1).

☑ Exemple :

```
LD    1     ;Charge l'état de l'entrée numérique 1 dans la pile
            ;logique.
LD    2     ;Charge l'état de l'entrée numérique 2 dans la pile
            ;logique.
ORLD        ;Ferme le bloc logique OR.
OUT   106   ;Le résultat du OR passe sur la sortie numérique 106.
```



OUT

OUT XXXX

MNÉMONIQUE : **OUT**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **09**

DESCRIPTION :

DÉCHARGE le dernier état de la pile logique dans l'opérande.
Décrémente la pile logique de 1 niveau (-1).

☑ Exemple :

```
LD   600   ;Charge l'état du relais interne 600 dans la pile logique.
OUT  106   ;L'état de l'entrée 1 passe sur la sortie numérique 106.
```

RELAIS 600 **RELAIS 106**

OUTNT

OUTNT XXXX

MNÉMONIQUE : **OUTNT**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **10**

DESCRIPTION :

DÉCHARGE le dernier état de la pile logique dans l'opérande après l'avoir complémenté.
Décrémente la pile logique de 1 niveau (-1).

☑ Exemple :

```
LD      1     ;Charge l'état de l'entrée numérique 1 dans la pile
              ;logique.
OUTNT   109   ;L'état complémenté de l'entrée 1 passe sur la
              ;sortie numérique 109.
```

RELAIS 1 **RELAIS 109**

OUTX

OUTX XXXX

MNÉMONIQUE : **OUTX**
OPÉRANDE XXXX : **Registre entier**
CODE INSTRUCTION : **16**

DESCRIPTION :

DÉCHARGE le dernier état de la pile logique dans le relais désigné par l'opérande.
L'opérande est le registre entier qui contient l'adresse du relais sur lequel s'effectue le déchargement.
Décrémente la pile logique de 1 niveau (-1).

☑ Exemple :

```
MOVCI   101
STOI    500   ;Stocke la donnée 101 dans le registre entier 500.
LD      1     ;Charge l'état de l'entrée 1.
OUTX    500   ;L'état de l'entrée 1 passe sur la sortie numérique
              ;101 (adresse contenue dans le registre 500).
```

SET

SET XXXX

MNÉMONIQUE : **SET**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **11**

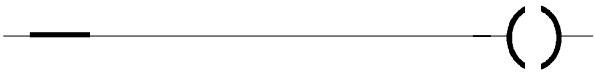
DESCRIPTION :

ACTIVE le relais désigné par l'opérande.
L'exécution de cette instruction ne doit être associée à aucune condition logique.
Un bit activé au moyen de cette instruction peut être désactivé par le résultat de toute autre instruction.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
SET  101   ;Activation de la sortie numérique 101.
```

RELAIS 101



RESET

RESET XXXX

MNÉMONIQUE : **RESET**
OPÉRANDE XXXX : **Relais interne, entrée ou sortie numérique**
CODE INSTRUCTION : **12**

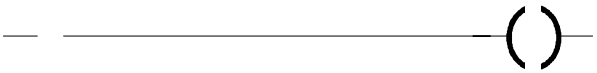
DESCRIPTION :

DÉSACTIVE le relais désigné par l'opérande.
L'exécution de cette instruction ne doit être associée à aucune condition logique.
Un bit désactivé au moyen de cette instruction peut être activé par le résultat de toute autre instruction.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
RESET    104    ;Désactivation de la sortie numérique 104.
```

RELAIS 104



JZ

JZ XXXX

MNÉMONIQUE : **JZ**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **93.**

DESCRIPTION :

Effectue un SAUT lorsque le dernier niveau de pile logique **est "0"**.
Si le niveau de la pile est "1", le programme passe à la ligne suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut.
Décrémente la pile logique de 1 niveau (-1).

☑ Exemple :

```
001   LD    8     ;Charge l'état de l'entrée 8 dans la pile logique.
002   JZ    100   ;Si l'état de l'entrée 8 est "0", le programme saute à la
                  ;ligne 100, si l'état de cette entrée est "1" il passe à la
                  ;ligne suivante (003).
003   LD    16
.....  .....  ....
.....  .....  ....
100   LD    610
```

JNZ

JNZ XXXX

MNÉMONIQUE : **JNZ**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **94.**

DESCRIPTION :

Effectue un SAUT lorsque le dernier niveau de pile logique est **"1"**.
Si le niveau de la pile est "0", le programme passe à la ligne suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut.
Décrémente la pile logique de 1 niveau (-1).

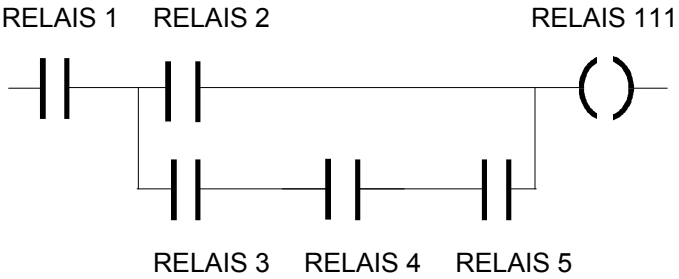
☑ Exemple :

```
001   LD    8     ;Charge l'état de l'entrée 8 dans la pile logique.
002   JNZ   100   ;Si l'état de l'entrée 8 est "1", le programme saute à la
                  ;ligne 100, si l'état de cette entrée est "0" il passe à la
                  ;ligne suivante (003).
003   LD    2
.....  .....  ....
.....  .....  ....
100   LD    16
```

☑ EXEMPLES :

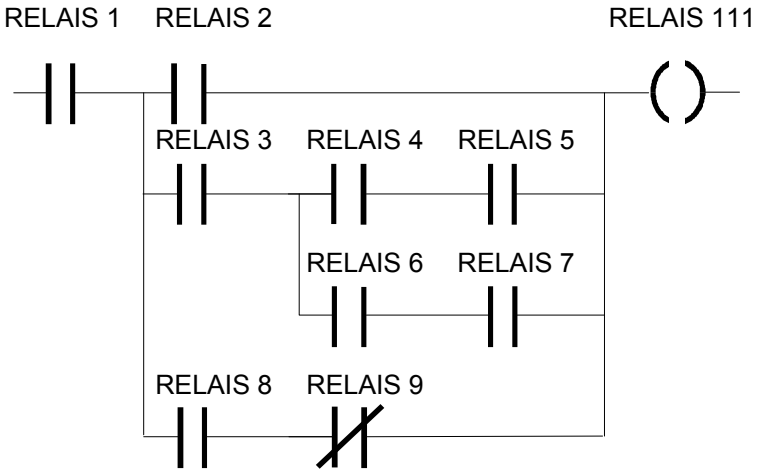
Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Élaboration du schéma à contacts suivant à partir d'une liste d'instructions.




```
LD      1     ;Charge l'état du relais 1 dans la pile logique.
LD      2     ;Charge l'état du relais 2 dans la pile logique.
LD      3     ;Charge l'état du relais 3 dans la pile logique.
AND     4     ;AND logique entre les relais 3 et 4.
AND     5     ;AND logique entre le résultat précédent et le
              ;relais 5.
ORLD          ;Ferme le bloc logique OR entre le résultat
              ;précédent et le relais 2.
ANDLD         ;Ferme le bloc logique en effectuant un AND entre
              ;le résultat précédent et le relais 1.
OUT     111   ;Le résultat précédent passe sur le relais 111.
END
```

B.- Réalisez le schéma à contacts suivant à partir d'une liste d'instructions




```
LD      1     ;Charge l'état du relais 1 dans la pile logique.
LD      2     ;Charge l'état du relais 2 dans la pile logique.
LD      3     ;Charge l'état du relais 3 dans la pile logique.
LD      4     ;Charge l'état du relais 4 dans la pile logique.
AND     5     ;AND logique entre les relais 4 et 5.
LD      6     ;Charge l'état du relais 6 dans la pile logique.
AND     7     ;AND logique entre les relais 6 et 7.
ORLD          ;Ferme le bloc logique OR entre les résultats
              ;des deux précédents AND.
ANDLD         ;Ferme le bloc logique en effectuant un AND entre
              ;le résultat précédent et le relais 3.
ORLD          ;Ferme le bloc logique OR entre le résultat
              ;précédent et le relais 2.
ANDLD         ;Ferme le bloc logique en effectuant un AND entre
              ;le résultat précédent et le relais 1.
LD      8     ;Charge l'état du relais 8 dans la pile logique.
ANDNT   9     ;AND entre l'entrée 8 et l'entrée 9 complémentée.
```

```
ORLD            ;Ferme le bloc logique OR entre le résultat
                ;précédent et le dernier état de la pile logique
                ;(résultat de la fermeture du bloc logique effectué
                ;au moyen d'un AND avec le relais 1).
OUT     111     ;Le résultat précédent passe sur le relais 111.
END
```

Pour les exemples ci-après, nous utiliserons un système basé sur l'assignation de numéros aux entrées et aux sorties (comme dans le cas des exemples précédents) auxquelles nous assignerons également des étiquettes.

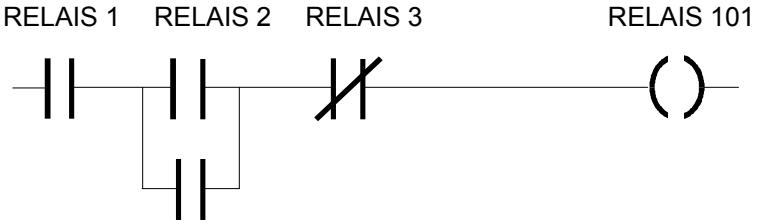
C.- MARCHE/ARRÊT AVEC ARRÊT D'URGENCE.

Nous avons tout d'abord assigné des numéros aux variables utilisées, dans le cas présent trois entrées et une sortie.

Le programme fonctionne de la manière suivante : supposons que le bouton d'arrêt d'urgence ne soit pas activé (souvenez-vous que le contact matériel du bouton d'arrêt d'urgence est un contact "repos") ; l'entrée 1 (relais 1) est activée à condition que le bouton d'arrêt d'urgence ne le soit pas. Lorsque vous appuyez sur le bouton de marche (entrée 3 = relais 3), le contacteur de mise en marche du moteur s'active (sortie 101 = relais 101) et, du fait du relais 101, reste verrouillé lorsque vous relâchez le bouton.
Pour désactiver la manœuvre, il vous faut donc appuyer sur le bouton d'arrêt (entrée 2 = relais 2). Le circuit s'ouvre, le contacteur du moteur (sortie 1 = relais 101) se déconnecte et le relais 101 se désactive.

La manœuvre est interrompue dès que vous appuyez sur le bouton d'arrêt d'urgence (entrée 1 = relais 1).

- Bouton d'arrêt d'urgence Entrée : 1
- Bouton d'arrêt Entrée : 2
- Bouton de marche Entrée : 3
- Contacteur de mise en marche du moteur Sortie : 101

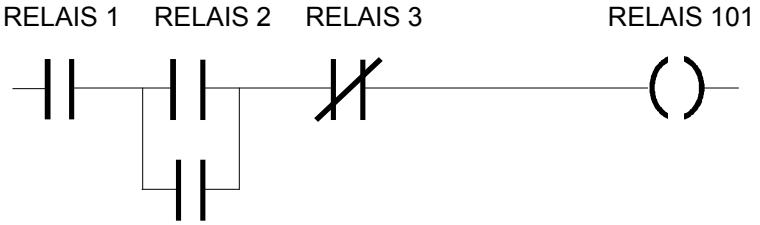


RELAIS 101

```
LD      1     ;Entrée arrêt d'urgence.
LD      3     ;Entrée marche.
OR      101   ;Entrée verrouillage.
ANDLD
ANDNT   2     ;Entrée arrêt.
OUT     101   ;Sortie activation/désactivation du contacteur du
              ;moteur.
```

Ci-après le même exemple, mais basé sur l'assignation d'étiquettes au lieu de numéros aux entrées et aux sorties.

- Bouton d'arrêt d'urgence Entrée : 1
- Bouton d'arrêt Entrée : 2
- Bouton de marche Entrée : 3
- Contacteur de mise en marche du moteur Sortie : 101



RELAIS 101

```
;DÉFINITION D'ÉTIQUETTES.
;Entrées numériques.
;
urgence     equ   1      ;Urgence.
arret       equ   2      ;Arrêt.
marche      equ   3      ;Marche.
;
;Sorties numériques.
;
moteur      equ   101    ;Contacteur moteur.
;
        LD          urgence     ;Entrée arrêt d'urgence.
        LD          marche      ;Entrée marche.
        OR          moteur      ;Verrouillage.
        ANDLD
        ANDNT       arret       ;Entrée arrêt.
        OUT         moteur      ;Sortie activation/désactivation du
                                ;contacteur du moteur.
```

FLANC

FLANC XXXX YYYY

MNÉMONIQUE : **FLANC**
OPÉRANDE XXXX : **Numéro de relais**
OPÉRANDE YYYY : **Registre interne de l'état du flanc (de 0 à 99).**
CODE INSTRUCTION : **36**

DESCRIPTION :

Détecte un flanc ascendant dans un registre de type relais.
L'opérande XXXX est le relais sur lequel le flanc est détecté.
L'opérande YYYY est le registre interne de flanc à utiliser (non accessible à l'utilisateur).
Retourne un "1" à la pile logique si un flanc ascendant a été détecté et un "0" dans le cas contraire.
Le contenu du relais désigné par l'opérande XXXX est stocké dans un registre interne d'état (opérande YYYY) chaque fois que la fonction FLANC est appelée. Si ce registre passe de l'état "0" à l'état "1", la fonction retourne un "1" à la pile logique. Dans le cas contraire, elle retourne un "0".
Incrémente la pile logique de 1 niveau (+1).

☑ Exemple :

```
        FLANC   0     1   ;Détecte la présence d'un flanc ascendant
                          ;sur l'entrée 0.
        JZ      ini1      ;En cas d'absence de flanc ascendant sur
                          ;l'entrée 0, saute à "ini1". Dans le cas
                          ;contraire, le programme se poursuit.
        INC     500   1   ;Incrémente le contenu du registre entier
                          ;500 de 1.
ini1    FLANC   1     2   ;Détecte la présence d'un flanc ascendant
                          ;sur l'entrée 1.
        JZ      ini2      ;En cas d'absence de flanc ascendant sur
                          ;l'entrée 2, saute à "ini2". Dans le cas
                          ;contraire, le programme se poursuit.
```

```
        INC     501   1     ;Incrémente le contenu du registre entier
                            ;501 de 1.
ini2    CLEAR                ;Efface la mémoire-tampon intermédiaire et
                            ;place le pointeur d'affichage sur la
                            ;première position.
        LOC     0           ;Place le pointeur sur la position 0.
        DISRI   500   2     ;Copie le contenu du registre entier 500 (à
                            ;deux chiffres) dans la mémoire-tampon
                            ;intermédiaire.
        LOC     16          ;Place le pointeur sur la position 16.
        DISRI   501   2     ;Copie le contenu du registre entier 501 (à
                            ;deux chiffres) dans la mémoire-tampon
                            ;intermédiaire.
        COM     0           ;Affiche le contenu de la mémoire-tampon
                            ;intermédiaire sur le LCD.
        END
```

Les équipements MIDA disposent d'instructions qui permettent de temporiser et de compter les impulsions (TIM, TIMR, CNT et CNTR).

Nous décrivons, ci-après, le mode de programmation de ces instructions.

✓ TEMPORISATEURS

Les temporisateurs des équipements MIDA sont des temporisateurs décrémentiels de connexion, c'est-à-dire que le temps sélectionné s'étant écoulé, ils transfèrent un état logique "1" à la pile ; le temps est exprimé en dixièmes de secondes.

Nous expliquerons tout d'abord ce qu'est une instruction de temporisation. Nous décrirons ensuite les types d'instructions dont disposent les équipements MIDA, puis verrons comment utiliser ces instructions.

Une instruction de temporisation possède la même fonction qu'un temporisateur matériel (hardware). Ce type d'instruction possède deux opérandes. Le premier indique le temporisateur sélectionné. Cette sélection peut être effectuée par assignation du numéro du temporisateur à utiliser ou par assignation d'une étiquette (voir exemples).

Le deuxième opérande indique la valeur numérique de la temporisation. Cette valeur est chargée dans le registre RAM de temporisation qui est le registre sur lequel s'exécute le décrément de la valeur numérique assignée, exprimée en dixièmes de seconde. Le registre du premier opérande indique le numéro du registre RAM dans lequel est chargée la valeur à temporiser dans le deuxième opérande, qui est uniquement accessible au temporisateur lors du rechargement, lorsque la condition appropriée se produit.

Lorsque le contenu du registre RAM appartenant au temporisateur est "0", cela signifie que le temps programmé s'est déjà écoulé et que la temporisation est terminée. Cela entraîne l'activation d'un "1" dans la pile logique. Si la temporisation n'est pas terminée, un "0" s'active dans la pile logique.

Ce registre de temporisation est un registre normal, accessible par le programme MIDA. Il peut donc être traité arithmétiquement

(addition, soustraction, etc.), être comparé à des constantes ou au contenu d'autres registres, être modifié, affiché, etc.

Souvenez-vous que la valeur de ce registre est décrémentielle : elle représente les dixièmes de seconde manquants pour atteindre la valeur 0 et non le temps écoulé. Pour déterminer la valeur de celui-ci, il vous suffit d'appliquer la formule arithmétique suivante :

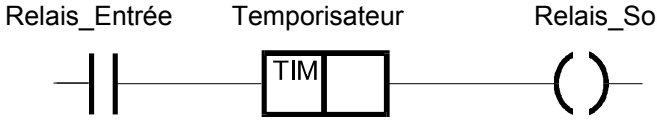
Valeur présélectionnée – Registre de temporisation = Temps écoulé.

La valeur numérique du deuxième opérande peut être assignée directement ou indirectement. Vous pouvez donc soit introduire directement la valeur numérique de la temporisation dans le deuxième opérande, soit lui assigner un registre entier dans lequel figure cette valeur. Dans le premier cas, il vous faut utiliser l'instruction TIM et dans le second, l'instruction TIMR.

Comme nous l'avons expliqué précédemment, l'emploi de l'instruction TIM implique la présélection d'une constante faisant partie intégrante du programme, ce qui limite la temporisation à une seule valeur, alors que l'emploi de l'instruction TIMR implique l'assignation du contenu d'un registre entier (16 bits), ce qui permet d'utiliser le même temporisateur pour différentes valeurs de temporisation à l'intérieur d'un programme.

Cela étant dit, nous allons vous expliquer comment utiliser une instruction de temporisation.

Pour exécuter une fonction de temporisation, il vous faut disposer de trois éléments, comme l'indique le schéma ci-dessous : un relais d'entrée, un temporisateur et un relais de sortie :



```
LD    Relais_Entrée
TIM   Temporisateur     Valeur présélectionnée
OUT   Relais_Sortie
```

Le relais d'entrée sert à autoriser ou non la mise en marche du temporisateur. Ce relais peut présenter deux états logiques : "0" et "1". L'état "1" correspond à l'état activé et l'état "0" à l'état désactivé. Un "1" logique permet d'activer le temporisateur.

C'est le temporisateur lui-même qui exécute la temporisation proprement dite après avoir été activé par le relais d'entrée, lui-même activé par un état logique "1". Le temporisateur ayant été activé, il continue à exécuter la temporisation, même si le relais responsable de son activation passe à "0". La sortie du temporisateur indique un "0" logique tant que la temporisation n'est pas terminée. Lorsque celle-ci est achevée, un "1" logique apparaît à la sortie du temporisateur.

Le relais de sortie est le relais sur lequel le temporisateur décharge l'état logique qui résulte de la temporisation ("1" si le temps s'est écoulé, "0" dans le cas contraire). Le relais de fin de temps peut, par exemple, être remplacé par un saut conditionnel, JZ ou JNZ, car ces instructions permettent également de décharger l'état logique transféré à la pile par le temporisateur.

Il existe également un relais de fin de temporisation qui "suit" l'état du temporisateur en cas d'interruption logicielle. Ce relais possède la même adresse que le temporisateur et son état est "0" si le registre du temporisateur est différent de "0" (temporisation non achevée) et "1" si le registre du temporisateur est "0" (temporisation terminée).

Les instructions de temporisation disponibles dans les équipements MIDA sont TIM et TIMR. Elles se différencient de la manière suivante : dans le premier cas (TIM), le temporisateur recharge la valeur présélectionnée contenue dans le programme sous la forme d'une constante, alors que dans le second (TIMR), ce rechargement s'effectue par l'intermédiaire de n'importe quel registre interne.

La valeur maximale de temporisation est de 3.276,7 secondes. Si cette valeur s'avère insuffisante, consultez le chapitre consacré aux compteurs.

Pour qu'une instruction de temporisation fonctionne, elle doit :

- avoir été rechargée lors de sa présélection.
- avoir été lancée par l'intermédiaire de l'exécution du code de programme.

Vous trouverez, ci-après, une description des différentes instructions de temporisation :

TIM

TIM XXXX YYYY

MNÉMONIQUE : **TIM**
OPÉRANDE XXXX : **Adresse du temporisateur**
OPÉRANDE YYYY : **Constante indiquant la présélection du temporisateur**
CODE INSTRUCTION : **13**

DESCRIPTION :

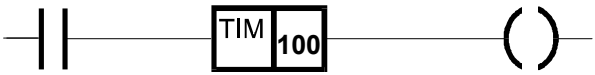
Charge un état "1" dans la pile logique après écoulement du TEMPS programmé.
L'opérande XXXX est l'adresse du temporisateur à utiliser.
L'opérande YYYY est la constante qui indique la valeur de temporisation présélectionnée, exprimée en dixièmes de seconde.
Si le contenu de la présélection du temporisateur est "0", la “sortie” de l'instruction TIM est "1".
Incrémente la pile logique de 1 niveau (+1).

RELAIS AUTORISATION	REGISTRE TEMPORISATION (RAM)	RELAIS FIN DE TEMPS ou RELAIS DE SORTIE	ÉTAT
"0"	Reg. = Présélection ≠ 0	"0"	RECHARGEMENT
"1"	Reg. = Reg. - 1 ≠ 0	"0"	Temporisation en cours
"1"	Reg. = Reg. - 1 = 0	"1"	TEMPS ÉCOULÉ
"0" ou "1"	Reg. = 0	"1"	Registre de temporisation : "0"

☑ Exemple :

```
LD    1          ;Charge l'état de l'entrée numérique 1 dans la pile
                 ;logique.
TIM   250  100   ;Temporisateur 250 programmé à 100 dixièmes
                 ;de seconde.
OUT   101        ;La sortie 101 s'active 10 secondes (100 dixièmes
                 ;de secondes) après l'activation de l'entrée 1.
                 ;(le relais interne 250 s'active également par
                 ;interruption).
```

RELAIS 1 **TEMPORISATEUR 250** **RELAIS 101**



TIMR

TIMR XXXX YYYY

MNÉMONIQUE : **TIMR**
OPÉRANDE XXXX : **Adresse du temporisateur**
OPÉRANDE YYYY : **Registre entier**
CODE INSTRUCTION : **21**

DESCRIPTION :

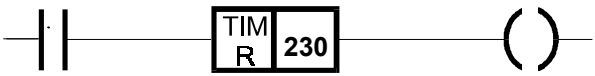
Charge un état "1" dans la pile logique après écoulement du TEMPS programmé.
L'opérande XXXX est l'adresse du temporisateur à utiliser.
L'opérande YYYY est le registre qui contient la constante indiquant la valeur de temporisation présélectionnée, exprimée en dixièmes de seconde.
Si le contenu du registre entier est "0", la “sortie” de l'instruction TIMR est "1".
Incrémente la pile logique de 1 niveau (+1).

RELAIS AUTORISATION	REGISTRE TEMPORISATION (RAM)	RELAIS DE FIN DE TEMPS ou DE SORTIE	ÉTAT
"0"	Reg. = Présélection ≠ 0	"0"	RECHARGEMENT
"1"	Reg. = Reg. - 1 ≠ 0	"0"	Temporisation en cours
"1"	Reg. = Reg. - 1 = 0	"1"	TEMPS ÉCOULÉ
"0" ou "1"	Reg. = 0	"1"	Registre de temporisation : "0"

☑ Exemple :

```
MOVCI   230         ;Charge la constante 230 dans la pile
                    ;arithmétique.
STOI    615         ;Stocke le contenu de la pile arithmétique
                    ;dans le registre entier 615.
LD      1           ;Charge l'état de l'entrée numérique 1 dans
                    ;la pile logique.
TIMR    280   615   ;Temporisateur 280 programmé avec le
                    ;contenu du registre entier 615 (230
                    ;dixièmes de seconde).
OUT     101         ;La sortie 101 s'active au bout 23 secondes
                    ;(230 dixièmes de seconde). Le relais
                    ;interne 280 s'active également par
                    ;interruption.
```

RELAIS 1 **TEMPORISATEUR 280** **RELAIS 101**



✓ COMPTEURS

Les compteurs des équipements MIDA sont des compteurs décrémentiels de connexion, c'est-à-dire que le comptage des impulsions s'étant écoulé conformément à la présélection, ils transfèrent un état logique "1" à la pile.

Nous expliquerons tout d'abord ce qu'est une instruction de compteur. Nous décrirons ensuite les types d'instructions dont disposent les équipements MIDA, puis verrons comment utiliser ces instructions.

Une instruction de compteur possède la même fonction qu'un compteur matériel (hardware). Ce type d'instruction possède deux opérandes. Le premier indique le compteur à utiliser, c'est-à-dire le compteur que vous allez sélectionner. Cette sélection peut être effectuée par assignation du numéro du compteur à utiliser ou par assignation d'une étiquette (voir exemples).

Le deuxième opérande indique la valeur numérique du compteur. Cette valeur est chargée dans le registre RAM du compteur qui est le registre sur lequel s'exécute le décrément de la valeur numérique assignée. Ce registre RAM constitue le premier opérande et la valeur qu'il contient est celle désignée par le deuxième opérande.

Lorsque le contenu du registre RAM appartenant au compteur est "0", cela signifie que le comptage programmé est achevé. Cela entraîne l'activation d'un "1" dans la pile logique. Si le comptage n'est pas terminé, un "0" s'active dans la pile logique.

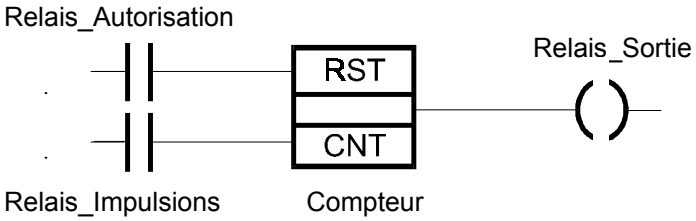
Ce registre de compteur est un registre normal, accessible par le programme MIDA. Il peut donc être traité arithmétiquement (addition, soustraction, etc.), être comparé à des constantes ou au contenu d'autres registres, être modifié, affiché, etc.

La valeur numérique du deuxième opérande est décrémentielle et peut être assignée directement ou indirectement. Vous pouvez donc soit introduire directement la valeur numérique du comptage à effectuer, soit lui assigner un registre entier dans lequel figure cette valeur. Dans le premier cas, il vous faut utiliser l'instruction CNT et dans le second, l'instruction CNTR.

Les instructions de compteurs dont disposent les équipements MIDA sont CNT et CNTR. Dans les grandes lignes, leur fonctionnement est tel que nous l'avons décrit précédemment. Comme nous l'avons expliqué ci-dessus, l'emploi de l'instruction CNT implique la présélection d'une constante faisant partie intégrante du programme, ce qui limite le comptage à une seule valeur, alors que l'emploi de l'instruction CNTR implique l'assignation du contenu d'un registre entier (16 bits), ce qui permet de faire varier le temps en fonction d'une introduction effectuée par l'intermédiaire du clavier, d'un résultat arithmétique ou d'un comptage d'impulsions, et de modifier chaque temporisation à volonté.

Cela étant dit, nous allons vous expliquer comment utiliser une instruction de compteur.

Pour exécuter une fonction de compteur, il vous faut disposer de quatre éléments, comme l'indique le schéma ci-dessous : un relais d'autorisation, un relais d'impulsions, un compteur et un relais de sortie.




```
LD    Relais_Autorisation
LD    Relais_Impulsions
CNT   Compteur présélection
OUT   Relais_Sortie
```

Avant d'examiner la fonction de chacun de ces éléments, nous aimerions ouvrir une parenthèse pour parler brièvement des différentes entrées dont dispose un compteur. Nous avons indiqué précédemment qu'une instruction de compteur est identique à un compteur matériel (hardware), c'est-à-dire qu'il comprend une entrée qui sert non seulement à autoriser ou non le comptage à effectuer,

mais permet aussi d'exécuter un RESET (remise à zéro) du compteur. C'est l'entrée identifiée par RST sur le schéma précédent. Elle agit comme un relais permettant de charger et de présélectionner le compteur.

La deuxième entrée du compteur est l'entrée d'impulsions. C'est par l'intermédiaire de cette entrée que s'effectue le comptage des impulsions. Elle est identifiée par CNT sur le schéma précédent. Le compteur possède enfin une sortie qui lui permet de transmettre un signal logique “1” ou “0” en fonction de l'état dans lequel il se trouve : état “1” si le comptage est terminé et état “0” s'il n'est pas achevé.

Cela étant dit, revenons au schéma précédent. Le relais d'autorisation d'entrée sert à autoriser ou non la mise en marche du compteur. Celui-ci peut présenter deux états logiques : "1" et "0". L'état "1" correspond à l'état activé et l'état "0" à l'état désactivé. Un "1" logique permet d'activer le compteur.

Le relais d'impulsions est le relais par lequel entrent les impulsions à déduire de la valeur présélectionnée jusqu'à ce que le résultat voulu (0) soit atteint.

C'est le compteur lui-même qui exécute le comptage proprement dit après avoir été activé par le relais d'entrée, lui-même activé par un état logique "1". La sortie du compteur indique un "0" logique tant que le comptage n'est pas terminé. Lorsque celui-ci est achevé, un "1" logique apparaît à la sortie du compteur.

Le relais de sortie est le relais sur lequel le compteur décharge l'état logique qui résulte du comptage ("1" si le comptage est terminé, "0" dans le cas contraire). Le relais de fin de temps peut, par exemple, être remplacé par un saut conditionnel, JZ ou JNZ, car ces instructions permettent également de décharger l'état logique transféré à la pile par le compteur.

Il existe également un relais de fin de comptage qui "suit" l'état du compteur en cas d'interruption logicielle. Ce relais possède la même adresse que le compteur et son état est "0" si le registre du compteur est différent de "0" (comptage non achevé) et "1" si le registre du compteur est "0" (comptage terminé).

Les instructions de compteurs disponibles dans les équipements MIDA sont CNT et CNTR. Elles se différencient de la manière

suivante: dans le premier cas (CNT), le compteur recharge la valeur présélectionnée contenue dans le programme sous la forme d'une constante, alors que dans le second (CNTR), ce rechargement s'effectue par l'intermédiaire de n'importe quel registre interne.

Nous aimerions également souligner qu'il est possible d'employer les compteurs en tant que temporisateurs en utilisant des relais de bases de temps en tant qu'entrées d'impulsions:

```
LD    400
LD    R_BASE          ;Le compteur dénombre les impulsions
                      ;de la base temps utilisée.
CNT   200   Compt_temps
OUT   401
```

Étant donné que le compteur dénombre les flancs ascendants de R_BASE, ce type de temporisation est deux fois plus précis que la temporisation de la base temps employée.

RELAIS D'IMPULSIONS	BASE TEMPS OBTENUE	DURÉE MAXIMALE POUVANT ÊTRE TEMPORISÉE
10 ms	20 ms	655,34 s (11 min)
100 ms	200 ms	6.553,4 s (109 min)
0,5 s	1 s	546 min (9 h)
1 s	2 s	1.092 min (18 h)
10 s	3 s	182 h (7,5 jours)
1 min	2 min	32.767 min (1.365 jours)

Il est donc possible de réaliser un comptage de temps à l'aide d'une base temps “fabriquée” par le programmeur. Pour obtenir des temporisations supérieures, vous pouvez enchaîner plusieurs compteurs, chacun d'entre eux se chargeant d'accumuler les heures, les jours, etc.

Vous trouverez, ci-après, la description des instructions de compteurs :

CNT

CNT XXXX YYYY

MNÉMONIQUE : **CNT**
OPÉRANDE XXXX : **Adresse du compteur**
OPÉRANDE YYYY : **Constante indiquant la présélection du compteur**
CODE INSTRUCTION : **14**

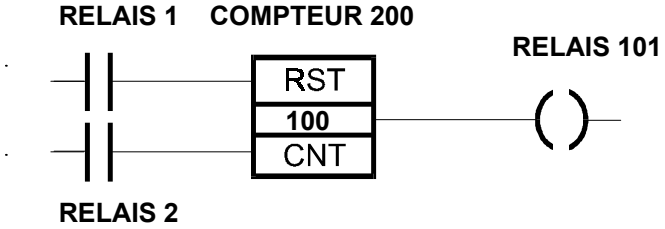
DESCRIPTION :

Charge un état "1" dans la pile logique après exécution du COMPTAGE programmé.
L'opérande XXXX est l'adresse du compteur à utiliser.
L'opérande YYYY est la constante qui indique la valeur de comptage présélectionnée, le nombre d'impulsions à dénombrer. Si la valeur présélectionnée est "0", la “sortie” de l'instruction CNT est "1".
Incrémente la pile logique de 1 niveau (+1).

RELAIS DÉBUT	ENTRÉE IMPULS.	REGISTRE COMPTAGE (RAM)	RELAIS DE SORTIE OU DE FIN DE COMPTAGE	ÉTAT
"1"	"X"	Reg. = Présélection	"0"	RECHARGEMENT
"0"	"0" , "1"	Reg. = Reg. - 1 ≠ 0	"0"	COMPTAGE EN COURS
"0"	"0" , "1"	Reg. = Reg. - 1 = 0	"1"	FIN DE COMPTAGE
"0" ou "1"	"X"	Reg. = 0	"1"	Registre de comptage : "0"

☑ Exemple :

```
LD    1          ;Charge l'état de l'entrée numérique 1 dans la
                 ;pile logique (début de comptage).
LD    2          ;Entrée des impulsions de comptage.
CNT   200  100   ;Comptage de 100 impulsions.
OUT   101        ;La sortie 101 s'active lorsque 100 impulsions ont
                 ;été dénombrées sur l'entrée 2 (le relais interne
                 ;200 s'active également par interruption).
```



CNTR

CNTR XXXX YYYY

MNÉMONIQUE : **CNTR**
OPÉRANDE XXXX : **Adresse du compteur**
OPÉRANDE YYYY : **Registre entier**
CODE INSTRUCTION : **22**

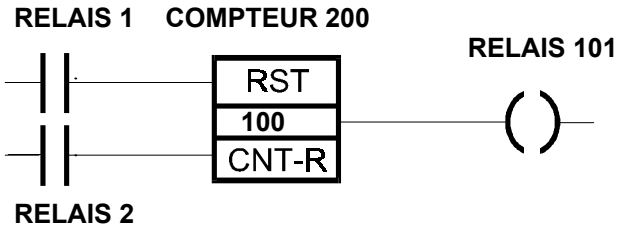
DESCRIPTION :

Charge un état "1" dans la pile logique après exécution du COMPTAGE programmé.
L'opérande XXXX est l'adresse du compteur à utiliser.
L'opérande YYYY est le registre entier qui contient la constante indiquant la valeur de comptage présélectionnée, le nombre d'impulsions à dénombrer.
Si le contenu du registre est "0", la “sortie” de l'instruction CNTR est "1".
Décrémente la pile logique de 1 niveau (-1).

RELAIS DÉBUT	ENTRÉE IMPULS.	REGISTRE COMPTAGE (RAM)	RELAIS DE SORTIE OU DE FIN DE COMPTAGE	ÉTAT
"1"	"X"	Reg. = Présélection	"0"	RECHARGEMENT
"0"	"0" , "1"	Reg. = Reg. - 1 $\neq$ 0	"0"	COMPTAGE EN COURS
"0"	"0" , "1"	Reg. = Reg. - 1 = 0	"1"	FIN DE COMPTAGE
"0" ou "1"	"X"	Reg. = 0	"1"	Registre de comptage : "0"

☑ Exemple :

```
MOVCI   100          ;Charge la constante 100 dans la pile
                     ;arithmétique.
STOI    500          ;Stocke le contenu de la pile arithmétique
                     ;dans le registre entier 500.
LD      1            ;Charge l'état de l'entrée numérique 1 dans
                     ;la pile logique (autorisation de comptage).
LD      2            ;Entrée des impulsions de comptage.
CNTR    200  500     ;Comptage de 100 impulsions (contenu du
                     ;registre entier 500) par l'entrée numérique
                     ;2. Il débute lorsque l'entrée 1 passe à
                     ;l'état "0".
OUT     101          ;La sortie 101 s'active lorsque 100
                     ;impulsions sont entrées (le relais interne
                     ;200 s'active également par interruption.
```



☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Mise en marche et remise à zéro d'un temporisateur

Cet exemple fonctionne de la manière suivante : l'entrée 1 ayant été activée, la sortie 104 ne passera pas à l'état "1" (ON) tant que le temps prédéterminé dans le temporisateur 250 ne se sera pas écoulé, à condition que l'entrée 2 ne soit pas activée. Si vous activez l'entrée 2, le temporisateur est remis à zéro et la sortie 104 passe à "0" si elle était préalablement à "1".

```
LD      1           ;Mise en marche du temporisateur.
OR      400
ANDNT   2           :Mise à zéro de la sortie 104 et
                    ;remise à zéro du temporisateur.
OUT     400         ;Sortie numérique contrôlée.
LD      400
TIM     250   10    ;Temps de maintien sur ON.
OUT     104         ;Sortie numérique temporisée.
```

B.- Mise en marche et remise à zéro d'un temporisateur avec stockage du temps présélectionné dans un registre entier.

L'exemple ci-dessous est similaire au précédent, mais au lieu d'utiliser l'instruction TIM et, en conséquence, d'introduire directement la valeur de temporisation (deuxième opérande), nous avons employé l'instruction TIMR, c'est-à-dire que la valeur de temporisation est stockée dans un registre interne de type entier désigné par le deuxième opérande, comme cela est décrit dans les commentaires qui accompagnent l'exemple de programme.

```
MOVCI   20          ;Charge la constante entière 20
                    ;dans la pile arithmétique.
STOI    300         ;Stocke cette constante dans le
                    ;registre entier 300.
```

```
LD      1           ;Mise en marche du temporisateur.
OR      400
ANDNT   2           :Mise à zéro de la sortie 104 et
                    ;remise à zéro du temporisateur.
OUT     400         ;Sortie numérique contrôlée.
LD      400
TIMR    250  300    ;Temps de maintien sur ON.
OUT     104         ;Sortie numérique temporisée.
```

C.- Générateur d'impulsions

La fréquence de sortie des impulsions varie en fonction des temps programmés dans les temporisateurs.

```
LD      1           ;Mise en marche du générateur
                    ;d'impulsions.
LD      400
ORLD
ANDNT   2           ;Arrêt du générateur d'impulsions.
OUT     400         ;Autorisation de mise en marche du
                    ;générateur d'impulsions.
LD      400
ANDNT   410         ;Temporisateur ON.
TIM     250  10     ;Temporisateur généré par l'état
                    ;ON.
OUT     405
LD      405
AND     104         ;Temporisateur OFF.
TIM     251  10     ;Temporisateur généré par l'état
                    ;OFF.
OUT     410
LD      405
OUT     104
```

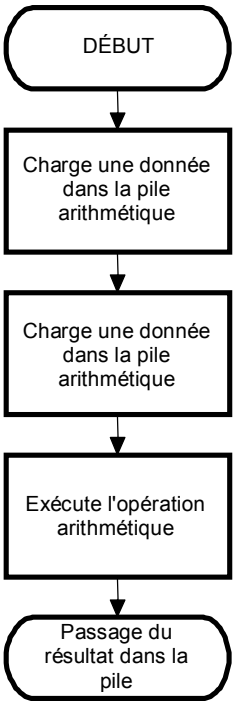
D.- Compteur de pièces

Il s'agit ici de compter 10 pièces puis, cela étant fait, d'indiquer que le comptage est achevé.

```
LD      1           ;Mise en marche du compteur.
OR      400
ANDNT   2           ;Arrêt du compteur et remise à zéro.
OUT     400         ;Autorisation compteur.
LDNT    400
LD      5           ;Entrée compteur.
CNT     200   10    ;Compteur avec valeur fixée à 10.
OUT     104         ;Sortie signal de fin de comptage.
```

Les fonctions arithmétiques pouvant être exécutées à l'aide des instructions dont disposent les équipements MIDA sont les suivantes :

- ***Addition, soustraction, multiplication et division, dans des formats entiers (registres de 16 bits) et à virgule flottante (registres de 32 bits).***
- ***Stockage et lecture du contenu de registres entiers et à virgule flottante.***
- ***Conversion de données entières en données à virgule flottante.***
- ***Conversion de données à virgule flottante en données entières, avec arrondi.***



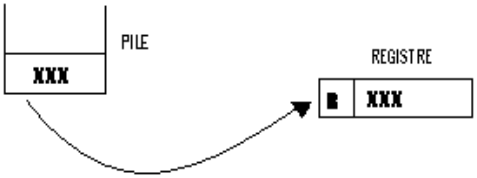
Le diagramme de flux décrit la manière dont les opérations arithmétiques sont exécutées.

Quelle que soit l'opération arithmétique à réaliser, les instructions manipulent les données entières et à virgule flottante à travers la pile arithmétique.

La manière d'opérer est la même quelle que soit l'opération : par notation polonaise avec accumulateur, en utilisant la pile arithmétique en tant que mémoire intermédiaire. Il est possible de travailler sur le contenu de registres ou à partir de constantes numériques, au format entier ou à virgule flottante, mais les données employées pour chaque opération doivent avoir le même format. Il est donc important de veiller à ce que le format des données des différentes opérations soit identique. Il existe pour ce faire des instructions qui permettent de convertir les données entières en données à virgule flottante et vice versa.

✓ STOCKAGE DES VALEURS DANS LES REGISTRES ET LECTURE

L'assignation des valeurs aux registres, entiers ou à virgule flottante, s'effectue par l'intermédiaire de la pile arithmétique.



Une donnée, entière ou à virgule flottante, doit être introduite dans la pile arithmétique pour être ensuite stockée dans un registre au format correspondant.

Pour assigner la donnée *cte* au registre *reg.*, vous devez exécuter l'un des codes de programme suivant, selon le format du registre concerné.

MOVCI cte STOI reg.		**reg. = cte**
SETRI	reg. cte	**reg. = cte**
MOVCF cte STOF reg.		**reg. = cte**

Tout stockage de donnée dans un registre implique la lecture préalable d'une constante ou du contenu d'un autre registre.

Les instructions de type MOVxx sont des instructions de lecture et les instructions de type STOxx sont des instructions de stockage.

L'instruction SETRI n'utilise pas la pile arithmétique.

Les instructions disponibles permettant de lire ou d'écrire des données dans les registres sont les suivantes :

MOVRI	MOVRF	MOVIF	STOF	SETRI
MOVCI	MOVCF	STOI	STOFX	
MOVIX	MOVFX	STOIX	STOFI	

✓ OPÉRATIONS ARITHMÉTIQUES SUR DES DONNÉES ENTIÈRES (16 bits avec signe)

Les opérations arithmétiques pouvant être exécutées à l'aide des instructions disponibles dans les équipements MIDA sont les suivantes :

- ***Addition***
- ***Soustraction***
- ***Multiplication***
- ***Division***

Toutes ces opérations s'effectuent avec la précision propre aux données entières de 16 bits avec signe, c'est-à-dire sans tenir compte des décimales, et avec une capacité de calcul de :

-32768 à 32767 ($2^{15} * 2^{15} -1$).

Donnée entière de 16 bits avec signe

S	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b
⇧	**OCTET DE DONNÉES**							**OCTET DE DONNÉES**							

(La flèche correspond au bit de signe).

Les instructions disponibles sont :

ADDI	SUBI	DIVI
ADDC	MULI	DIVC
SUBI	MULC	INC

Ces instructions permettent de réaliser des opérations entre registres entiers, entre constantes ou entre registres et constantes, comme le montre le tableau ci-dessous :

MOVRI A MOVRI B ADDI STOI C	**reg. C = reg. A + reg. B**
MOVRI A MOVCI B SUBI STOI C	**reg. C = reg. A - B**
MOVCI A MOVRI B MULI STOI C	**reg. C = A * reg. B**
MOVCI A MOVCI B DIVI STOI C	**reg. C = A / B**
MOVRI A MOVRI B MOVRI C ADDI SUBI STOI D	**reg. D = A+B-C**

Lorsque les opérations portent sur le contenu d'un registre et une constante, vous pouvez utiliser les instructions suivantes pour "économiser" des lignes de programme :

MOVRI A ADDC B STOI C	**reg. C = reg. A + B**
MOVRI A SUBC B STOI C	**reg. C = reg. A - B**
MOVRI A MULC B STOI C	**reg. C = reg. A * B**
MOVRI A DIVC B STOI C	**reg. C = reg. A / B**
MOVRI A MOVRI B ADDC C DIVI STOI D	**reg. C = reg. B + C / A**

✓ OPÉRATIONS ARITHMÉTIQUES SUR DES DONNÉES À VIRGULE FLOTTANTE (32 bits IEEE).

Les opérations arithmétiques pouvant être exécutées à l'aide des instructions disponibles dans les équipements MIDA sont les suivantes :

- ***Addition***
- ***Soustraction***
- ***Multiplication***
- ***Division***

Toutes ces opérations sont effectuées avec la précision propre aux données à virgule flottante de 32 bits au format IEEE, mais avec une capacité de représentation à l'écran et de transmission (via les ports de communication) de 16 chiffres significatifs.

La capacité de calcul va de **$3.4*10^{38}$ à $-3.4*10^{38}$** (avec un maximum de 16 chiffres affichables).

Donnée à virgule flottante de 32 bits IEEE

S	Exposant	Mantisse
⇧	**EXPOSANT (8 bits)**	**MANTISSE (23 bits)**

(La flèche correspond au bit de signe)

Les instructions dont disposent les équipements MIDA pour ce type d'opération sont les suivantes :

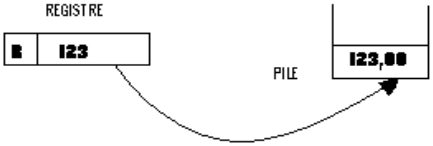
ADDF SUBF MULF DIVF

Ces opérations peuvent s'appliquer au contenu de registres, à des constantes ou au contenu de registres et à des constantes, comme le montre le tableau ci-dessous :

MOVRF A MOVRF B ADDF STOF C	**reg. C = reg. A + reg. B**
MOVRF A MOVCF B SUBF STOF C	**reg. C = reg. A – B**
MOVCF A MOVRF B MULF STOF C	**reg. C = A * reg. B**
MOVCF A MOVCF B DIVF STOF C	**reg. C = A / B**
MOVCF A MOVCF B MOVCF C MULF DIV STOI D	**reg. D = B * C / A**

✓ CONVERSION DES DONNÉES

L'équipement dispose d'instructions permettant de convertir les données entières en données à virgule flottante et vice versa.



La conversion des données entières en données à virgule flottante s'effectue par l'intermédiaire de la pile arithmétique, grâce à l'instruction MOVIF.

Les instructions de conversion passent le contenu du registre entier concerné dans la pile arithmétique et convertissent les données entières en données à virgule flottante de 32 bits, ce qui permet de travailler avec la précision de calcul souhaitée.

La conversion des données à virgule flottante en données entières s'effectue grâce à l'instruction STOFI, qui transfère les données à virgule flottante de la pile arithmétique vers un registre entier où elles sont stockées.

Ces données sont arrondies par excès ou par défaut, selon leur valeur :

Jusqu'à XX,4999	Par défaut	XX
À partir de XX,5000	Par excès	XX + 1

La donnée 23,34 sera arrondie à 23, et 23,68 à 24.

Les données à virgule flottante sont ainsi converties en données entières de 16 bits, ce qui permet de travailler avec la précision de calcul souhaitée.

✓ LISTE DES INSTRUCTIONS

Avant de commencer à détailler chacune des instructions qui composent ce groupe, nous vous fournissons, ci-dessous, un tableau récapitulatif de ces instructions.

MOVRI	**MOVRF**	**ADDI**	**ADDF**
MOVCI	**MOVCF**	**SUBI**	**SUBF**
MOVIX	**MOVFX**	**MULI**	**MULF**
SETRI		**DIVI**	**DIVF**
		ADDC	
STOI	**STOF**	**SUBC**	
STOIX	**STOFX**	**MULC**	
MOVIF	**STOFI**	**DIVC**	**INC**

Vous trouverez, ci-après, une description détaillée de ces instructions :

MOVRI

MOVRI XXXX

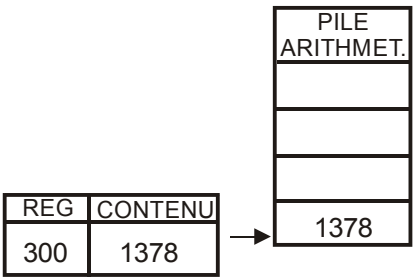
MNÉMONIQUE : **MOVRI**
OPÉRANDE XXXX : **Registre entier de 16 bits**
CODE INSTRUCTION : **33**

DESCRIPTION :

CHARGE le contenu du registre entier dans la pile arithmétique.
L'opérande est l'adresse du registre entier dont le contenu est transféré à la pile arithmétique.
Incrémente la pile arithmétique de 1 niveau (+1).

☑ Exemple :

```
MOVRI    300    ;Charge le contenu du registre entier 300 (par
                ;exemple 1378) dans la pile arithmétique.
```



MOVRI

MOVCI

MOVCI XXXX

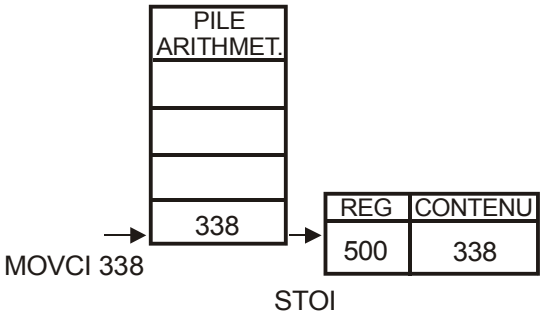
MNÉMONIQUE : **MOVCI**
OPÉRANDE XXXX : **Constante entière de 16 bits**
CODE INSTRUCTION : **32**

DESCRIPTION :

CHARGE une constante entière dans la pile arithmétique.
L'opérande est la constante entière qui est introduite dans le programme sous forme décimale et est transférée à la pile arithmétique.
Incrémente la pile arithmétique de 1 niveau (+1).

☑ Exemple :

```
MOVCI   338   ;Charge la constante entière 338 dans la pile
              ;arithmétique.
STOI    500   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 500.
```



STOI

STOI XXXX

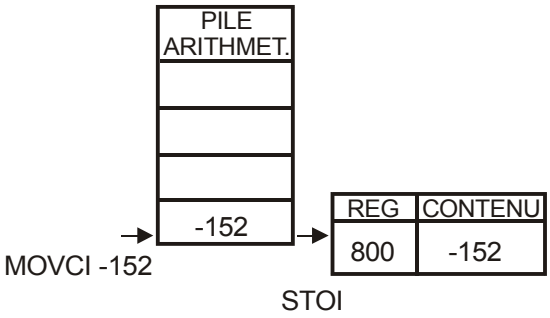
MNÉMONIQUE : **STOI**
OPÉRANDE XXXX : **Registre entier de 16 bits**
CODE INSTRUCTION : **34**

DESCRIPTION :

STOCKE la dernière donnée de la pile arithmétique dans un registre entier.
L'opérande est le registre entier dans lequel la dernière donnée de la pile arithmétique est stockée.
Décrémente la pile arithmétique de 1 niveau (-1).

☑ Exemple :

```
MOVCI   -152   ;Charge la constante -152 dans la pile
               ;arithmétique.
STOI    800    ;Stocke le contenu de la pile arithmétique dans le
               ;registre entier 800.
```



SETRI

SETRI XXXX YYYY

MNÉMONIQUE : **SETRI**
OPÉRANDE XXXX : **Registre entier de 16 bits**
OPÉRANDE YYYY : **Constante entière**
CODE INSTRUCTION : **35**

DESCRIPTION :

STOCKE la constante désignée par le deuxième opérande dans le registre entier du premier opérande.
L'opérande XXXX est l'adresse du registre entier dans lequel la donnée est chargée.
L'opérande YYYY est la constante entière qui est stockée dans le registre entier désigné par l'opérande XXXX.
Le niveau de la pile arithmétique reste inchangé (0).

☑ Exemple :

```
SETRI     300   3200  ;Stocke la donnée 3200 dans le registre
                      ;entier 300.
```

REG	CONTENU
300	3200

SETRI

MOVIX

MOVIX XXXX

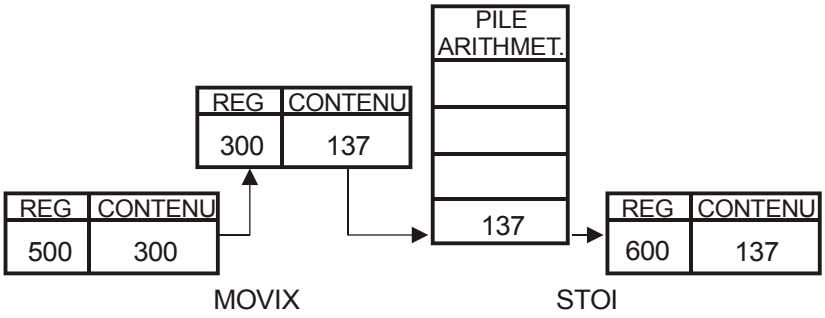
MNÉMONIQUE : **MOVIX**
OPÉRANDE XXXX : **Registre entier de 16 bits**
CODE INSTRUCTION : **30**

DESCRIPTION :

CHARGE, dans la pile arithmétique, le contenu du registre entier désigné.
L'opérande est le registre entier qui contient l'adresse du registre entier dont le contenu doit être transféré à la pile arithmétique.
Incrémente la pile arithmétique de 1 niveau (+1).

☑ Exemple :

```
SETRI   300   137   ;Stocke la constante 137 dans le registre
                    ;entier 300.
SETRI   500   300   ;Stocke la constante 300 dans le registre
                    ;entier 500.
MOVIX   500         ;Charge le contenu du registre entier
                    ;désigné par le registre entier 500 (reg.
                    ;300), c'est-à-dire la donnée 137, dans la
                    ;pile arithmétique.
STOI    600         ;Stocke le contenu de la pile arithmétique
                    ;dans le registre entier 600.
```



STOIX

STOIX XXXX

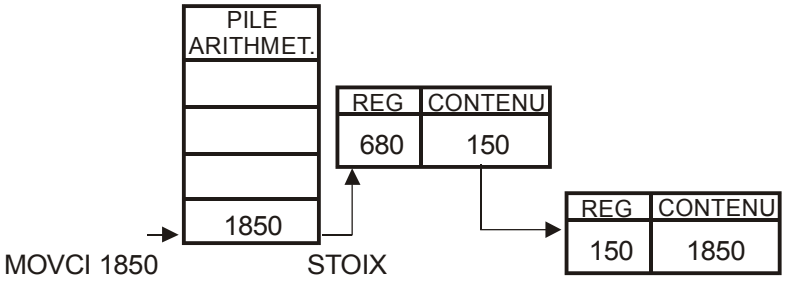
MNÉMONIQUE : **STOIX**
OPÉRANDE XXXX : **Registre entier de 16 bits**
CODE INSTRUCTION : **31**

DESCRIPTION :

STOCKE la dernière donnée de la pile arithmétique dans le registre entier désigné.
L'opérande est le registre entier qui contient l'adresse du registre entier dans lequel la dernière donnée de la pile arithmétique est stockée.
Décrémente la pile arithmétique de 1 niveau (-1).

☑ Exemple :

```
SETRI   680   150   ;Stocke la constante 150 dans le registre
                    ;entier 680.
MOVCI   1850        ;Charge la donnée 1850 dans la pile
                    ;arithmétique.
STOIX   680         ;Stocke le contenu de la pile arithmétique
                    ;dans le registre désigné par le registre
                    ;entier 680, c'est-à-dire dans le registre
                    ;entier 150.
```



MOVRF

MOVRF XXXX

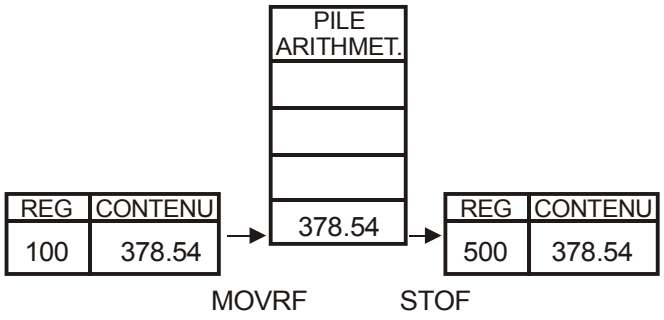
MNÉMONIQUE : **MOVRF**
OPÉRANDE XXXX : **Registre de 32 bits à virgule flottante**
CODE INSTRUCTION : **27**

DESCRIPTION :

CHARGE le contenu d'un registre à virgule flottante dans la pile arithmétique.
L'opérande est l'adresse du registre à virgule flottante dont le contenu est transféré à la pile arithmétique.
Incrémente la pile arithmétique de 2 niveaux (+2).

☑ Exemple :

```
MOVRF   100   ;Charge le contenu du registre à virgule flottante
              ;100, par exemple 378.54, dans la pile
              ;arithmétique.
STOF    500   ;Stocke le contenu de la pile arithmétique dans le
              ;registre à virgule flottante 500.
```



STOF

STOF XXXX

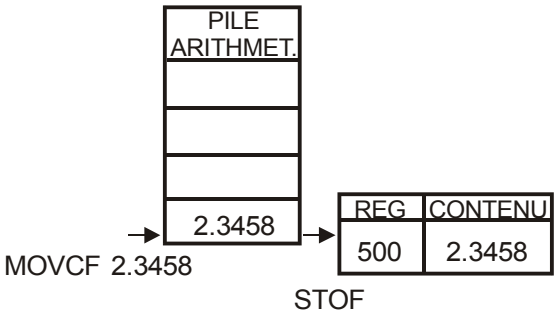
MNÉMONIQUE : **STOF**
OPÉRANDE XXXX : **Registre de 32 bits à virgule flottante**
CODE INSTRUCTION : **28**

DESCRIPTION :

STOCKE la dernière donnée de la pile arithmétique dans un registre à virgule flottante.
L'opérande est l'adresse du registre à virgule flottante dans lequel la dernière donnée de la pile arithmétique est stockée.
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
MOVCF   2.3458   ;Charge la constante 2.3458 dans la pile
                 ;arithmétique.
STOF    500      ;Stocke le contenu de la pile arithmétique
                 ;dans le registre à virgule flottante 500.
```



MOVCF

MOVCF XXXX

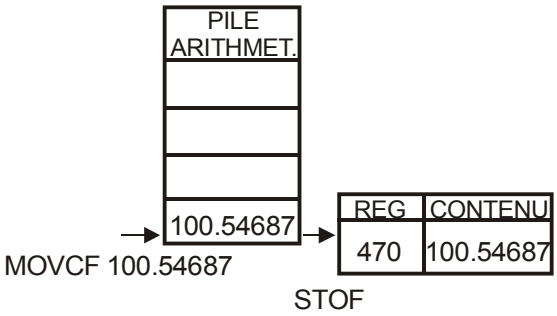
MNÉMONIQUE : **MOVCF**
OPÉRANDE XXXX : **Constante à virgule flottante**
CODE INSTRUCTION : **26**

DESCRIPTION :

CHARGE une constante à virgule flottante dans la pile arithmétique. L'opérande est la constante à virgule flottante introduite dans le programme sous forme décimale et transférée à la pile arithmétique. Incrémente la pile arithmétique de 2 niveaux (+2).

☑ Exemple :

```
MOVCF   100.54687   ;Charge la constante à virgule flottante
                    ;100.54687 dans la pile arithmétique.
STOF    470         ;Stocke le contenu de la pile arithmétique
                    ;dans le registre à virgule flottante 470.
```



MOVFX

MOVFX XXXX

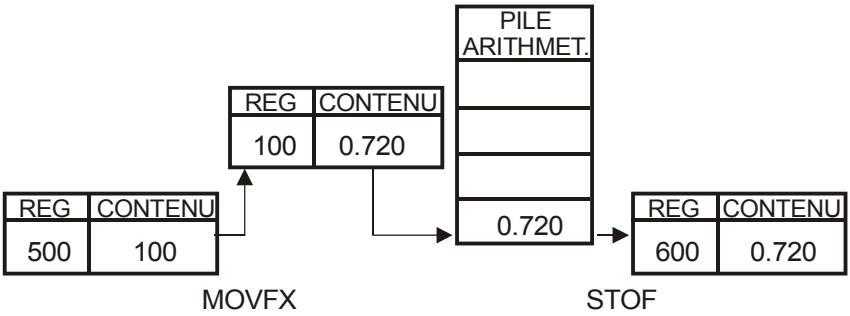
MNÉMONIQUE : **MOVFX**
OPÉRANDE XXXX : **Registre entier de 16 bits**
CODE INSTRUCTION : **23**

DESCRIPTION :

CHARGE, dans la pile arithmétique, le contenu du registre à virgule flottante désigné.
L'opérande est le registre entier qui contient l'adresse du registre à virgule flottante dont le contenu doit être transféré à la pile arithmétique.
Incrémente la pile arithmétique de 2 niveaux (+2).

☑ Exemple :

```
MOVCF   0.720        ;Charge la constante à virgule flottante
                     ;0.720 dans la pile arithmétique.
STOF    100          ;Stocke le contenu de la pile arithmétique
                     ;dans le registre à virgule flottante 100.
SETRI   500   100    ;Stocke la constante 100 dans le registre
                     ;entier 500.
MOVFX   500          ;Charge le contenu du registre entier
                     ;désigné par le registre à virgule flottante
                     ;100 (reg. 500), c'est-à-dire la donnée
                     ;0.720, dans la pile arithmétique.
STOF    600          ;Stocke le contenu de la pile arithmétique
                     ;dans le registre à virgule flottante 600.
```



STOFX

STOFX XXXX

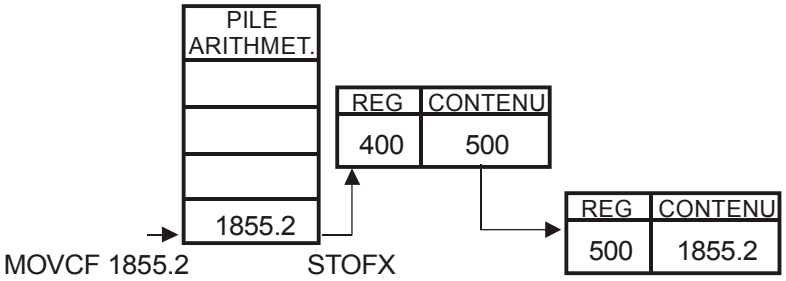
MNÉMONIQUE : **STOFX**
OPÉRANDE XXXX : **Registre entier de 16 bits**
CODE INSTRUCTION : **24**

DESCRIPTION :

STOCKE la dernière donnée de la pile arithmétique dans le registre à virgule flottante désigné.
L'opérande est le registre entier qui contient l'adresse du registre à virgule flottante dans lequel la dernière donnée de la pile arithmétique est stockée.
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
SETRI    400   500   ;Stocke la constante 500 dans le registre
                     ;entier 400.
MOVCF    18558.2     ;Charge la constante à virgule flottante
                     ;18558.2 dans la pile arithmétique.
STOFX    400         ;Stocke le contenu de la pile arithmétique
                     ;dans le registre à virgule flottante désigné
                     ;par le registre entier 400, c'est-à-dire dans
                     ;le registre 500.
```



ADDI

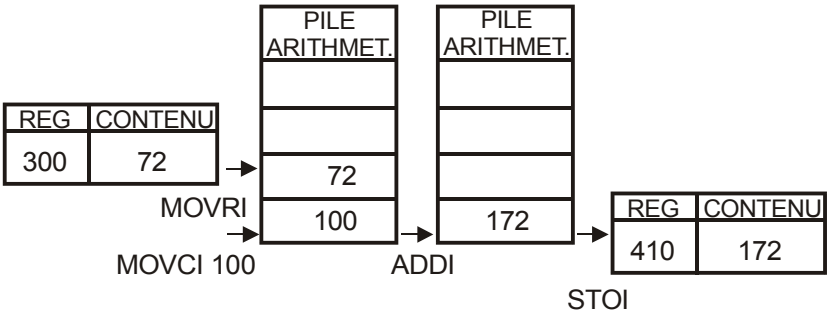
MNÉMONIQUE : **ADDI**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **40**

DESCRIPTION :

FAIT LA SOMME des deux dernières données entières de la pile arithmétique et place le résultat dans la pile arithmétique.
Décrémente la pile arithmétique de 1 niveau (-1).

☑ Exemple :

```
MOVCI   100   ;Charge la constante 100 dans la pile
              ;arithmétique.
MOVRI   300   ;Charge le contenu du registre entier 300 (par
              ;exemple 72) dans la pile arithmétique.
ADDI          ;Fait la somme des deux dernières données
              ;contenues dans la pile arithmétique et place le
              ;résultat, c'est-à-dire la donnée 172, dans cette
              ;même pile.
STOI    410   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 410 (résultat de la somme).
```



SUBI

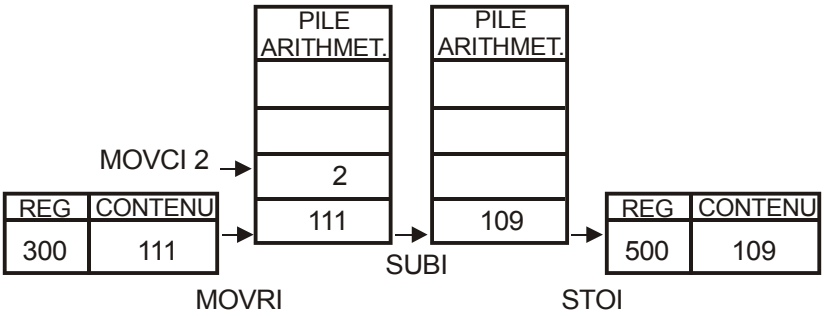
MNÉMONIQUE : **SUBI**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **41**

DESCRIPTION :

FAIT LA DIFFÉRENCE entre les deux dernières données entières de la pile arithmétique et place le résultat dans la pile arithmétique.
Le plus petit terme est la dernière donnée de la pile arithmétique. Le plus grand terme est la précédente.
Décrémente la pile arithmétique de 1 niveau (-1).

☑ Exemple :

```
MOVRI   300   ;Charge le contenu du registre entier 300 (par
              ;exemple 111) dans la pile arithmétique.
MOVCI   2     ;Charge la constante 2 dans la pile arithmétique.
SUBI          ;Fait la différence entre les deux dernières
              ;données contenues dans la pile arithmétique et
              ;place le résultat, c'est-à-dire la donnée 109, dans
              ;cette même pile.
STOI    500   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 500 (résultat de la différence).
```



MULI

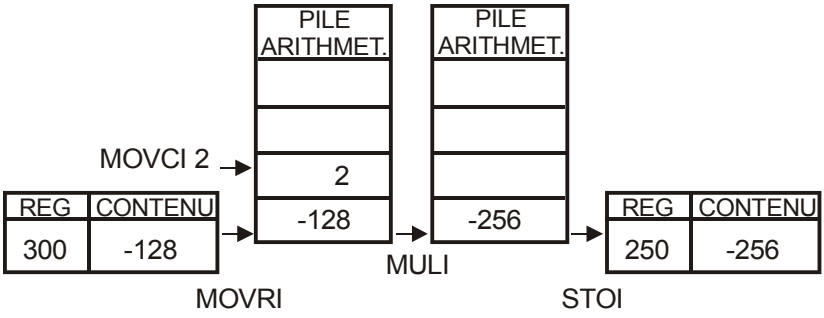
MNÉMONIQUE : **MULI**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **42**

DESCRIPTION :

MULTIPLIE les deux dernières données entières de la pile arithmétique entre elles et place le résultat dans la pile arithmétique. Décrémente la pile arithmétique de 1 niveau (-1).

☑ Exemple :

```
MOVRI   300   ;Charge le contenu du registre entier 300 (par
              ;exemple -128) dans la pile arithmétique.
MOVCI   2     ;Charge la constante 2 dans la pile arithmétique.
MULI          ;Multiplie les deux dernières données contenues
              ;dans la pile arithmétique entre elles et place le
              ;résultat, c'est-à-dire -256, dans cette même pile.
STOI    250   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 250 (résultat de la multiplication).
```



DIVI

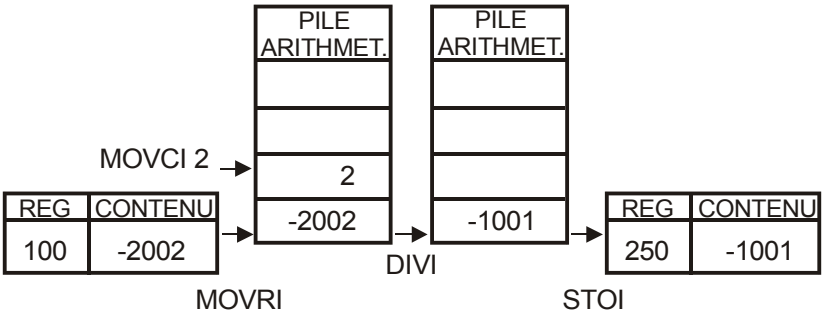
MNÉMONIQUE : **DIVI**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **43**

DESCRIPTION :

DIVISE les deux dernières données entières de la pile arithmétique entre elles et place le résultat dans la pile arithmétique.
Le diviseur est le dernier état de la pile arithmétique. Le dividende est le précédent.
Décrémente la pile arithmétique de 1 niveau (-1).

☑ Exemple :

```
MOVRI   100   ;Charge le contenu du registre entier 100, par
              ;exemple -2002, dans la pile arithmétique.
MOVCI   2     ;Charge la constante 2 dans la pile arithmétique.
DIVI          ;Divise les deux dernières données contenues
              ;dans la pile arithmétique entre elles et place le
              ;résultat, c'est-à-dire -1001, dans cette même pile.
STOI    250   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 250 (résultat de la division).
```



ADDC

ADDC XXXX

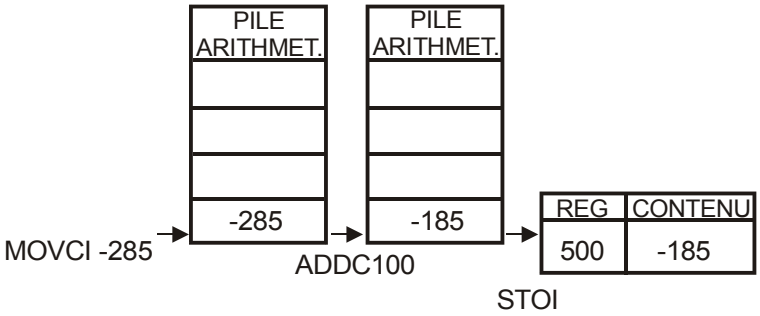
MNÉMONIQUE : **ADDC**
OPÉRANDE XXXX : **Constante entière**
CODE INSTRUCTION : **45**

DESCRIPTION :

AJOUTE la constante de l'opérande à la dernière donnée de la pile arithmétique.
L'opérande est la constante entière introduite dans le programme sous forme décimale et ajoutée à la dernière donnée de la pile arithmétique.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
MOVCI   -285   ;Charge la constante -285 dans la pile
               ;arithmétique
ADDC    100    ;Ajoute la constante 100 à la dernière donnée de
               ;la pile arithmétique et place le résultat, c'est-à-
               ;dire -185, dans la pile arithmétique.
STOI    255    ;Stocke le contenu de la pile arithmétique dans le
               ;registre entier 255 (résultat de la somme).
```



SUBC

SUBC XXXX

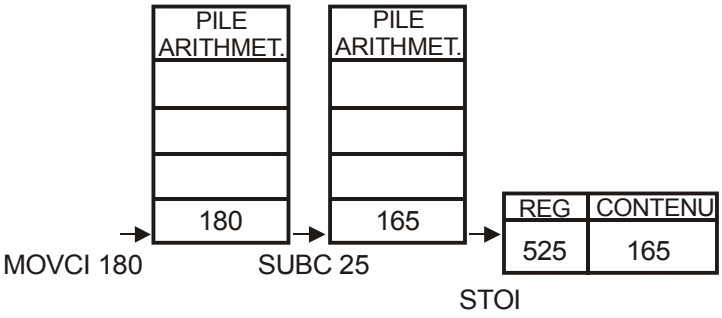
MNÉMONIQUE : **SUBC**
OPÉRANDE XXXX : **Constante entière**
CODE INSTRUCTION : **46**

DESCRIPTION :

SOUSTRAIT la constante de l'opérande de la dernière donnée de la pile arithmétique.
L'opérande est la constante entière (le plus petit terme) introduite dans le programme sous forme décimale et soustraite de la dernière donnée de la pile arithmétique.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
MOVCI   180   ;Charge la constante 180 dans la pile
              ;arithmétique.
SUBC    25    ;Soustrait la constante 25 de la dernière donnée
              ;de la pile arithmétique et place le résultat, c'est-à-
              ;dire 165, dans la pile arithmétique.
STOI    525   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 525 (résultat de la soustraction).
```



MULC

MULC XXXX

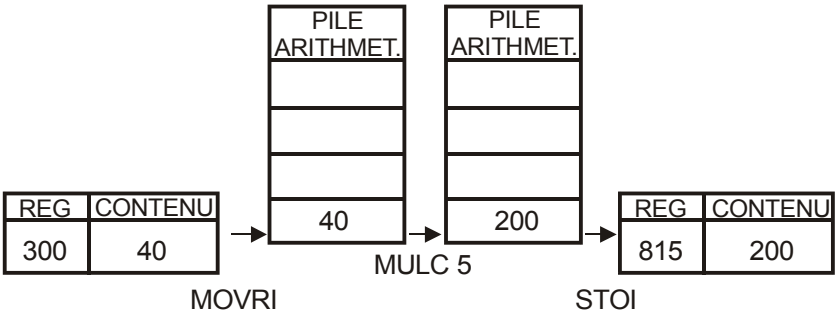
MNÉMONIQUE : **MULC**
OPÉRANDE XXXX : **Constante entière**
CODE INSTRUCTION : **47**

DESCRIPTION :

MULTIPLIE la dernière donnée de la pile arithmétique avec la constante de l'opérande.
L'opérande est la constante entière introduite dans le programme sous forme décimale et multipliée par la dernière donnée de la pile arithmétique.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
MOVRI   300   ;Charge le contenu du registre entier 300, par
              ;exemple 40, dans la pile arithmétique.
MULC    5     ;Multiplie la dernière donnée de la pile
              ;arithmétique par la constante 5 et place le
              ;résultat, c'est-à-dire 200, dans cette même pile.
STOI    815   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 815 (résultat de la multiplication).
```



DIVC

DIVC XXXX

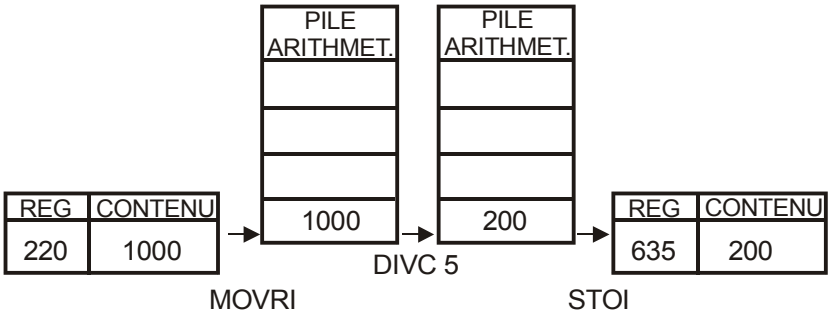
MNÉMONIQUE : **DIVC**
OPÉRANDE XXXX : **Constante entière**
CODE INSTRUCTION : **48**

DESCRIPTION :

DIVISE la dernière donnée de la pile arithmétique par la constante de l'opérande.
L'opérande est la constante entière introduite dans le programme sous forme décimale et le diviseur de la dernière donnée de la pile arithmétique.
Le niveau de la pile logique reste inchangé (0).

☑ Exemple :

```
MOVRI   220   ;Charge le contenu du registre entier 220, par
              ;exemple 1000, dans la pile arithmétique.
DIVC    5     ;Divise la dernière donnée de la pile arithmétique
              ;par la constante 5 et place le résultat, c'est-à-dire
              ;200, dans cette même pile arithmétique.
STOI    635   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 635 (résultat de la division).
```



INC

INC XXXX YYYY

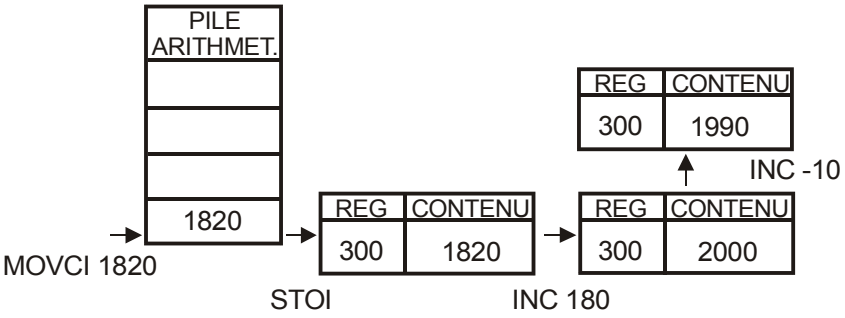
MNÉMONIQUE : **INC**
OPÉRANDE XXXX : **Registre entier de 16 bits**
OPÉRANDE YYYY : **Constante entière**
CODE INSTRUCTION : **44**

DESCRIPTION :

INCRÉMENTE ou DÉCRÉMENTE le contenu d'un registre entier d'une quantité constante entière.
L'opérande XXXX est l'adresse du registre entier à incrémenter ou à décrémenter.
L'opérande YYYY est la quantité incrémentée au registre entier ou décrémentée de celui-ci, selon son signe : positif ou négatif.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
MOVCI   1820        ;Charge la constante 1820 dans la pile
                    ;arithmétique.
STOI    300         ;Stocke le contenu de la pile arithmétique
                    ;dans le registre 300.
INC     300   180   ;Incrémente le contenu du registre entier
                    ;300 de 180, et le fait donc passer à 2000.
INC     300   -10   ;Décrémente le contenu du registre entier
                    ;300 de 10, et le fait donc passer à 1990.
```



ADDF

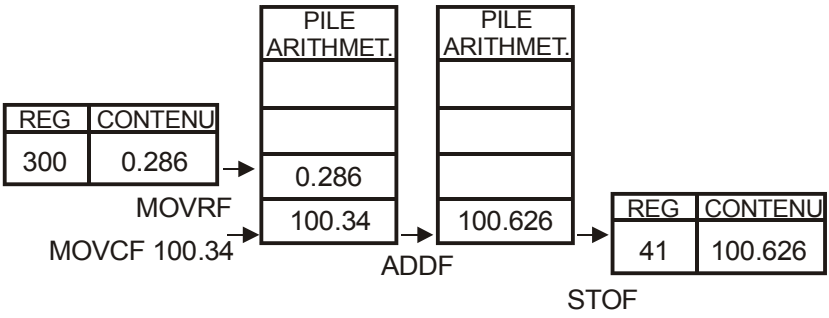
MNÉMONIQUE : **ADDF**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **49**

DESCRIPTION :

FAIT LA SOMME des deux dernières données à virgule flottante de la pile arithmétique et place le résultat dans la pile arithmétique. Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
MOVCF   100.34   ;Charge la constante à virgule flottante
                 ;100.34 dans la pile arithmétique.
MOVRF   300      ;Charge le contenu du registre à virgule
                 ;flottante 300, par exemple 0.286, dans la
                 ;pile arithmétique.
ADDF             ;Fait la somme des deux dernières
                 ;données contenues dans la pile
                 ;arithmétique et place le résultat, c'est-à-
                 ;dire 100.626, dans cette même pile.
STOF    41       ;Stocke le contenu de la pile arithmétique
                 ;dans le registre à virgule flottante 41
                 ;(résultat de la somme).
```



SUBF

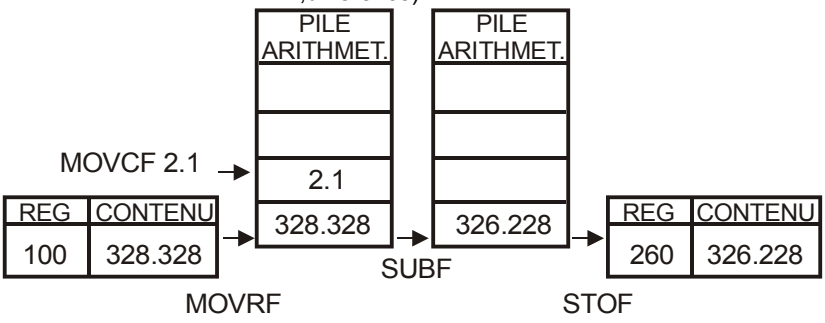
MNÉMONIQUE : **SUBF**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **50**

DESCRIPTION :

FAIT LA DIFFÉRENCE entre les deux dernières données à virgule flottante de la pile arithmétique et place le résultat dans la pile arithmétique.
Le plus petit terme est la dernière donnée de la pile arithmétique. Le plus grand terme est la précédente.
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
MOVRF    100   ;Charge le contenu du registre à virgule flottante
               ;100, par exemple 328.328, dans la pile
               ;arithmétique.
MOVCF    2.1   ;Charge la constante 2.1 dans la pile arithmétique.
SUBF           ;Fait la différence entre les deux dernières
               ;données contenues dans la pile arithmétique et
               ;place le résultat, c'est-à-dire 326.228, dans cette
               ;même pile.
STOF     260   ;Stocke le contenu de la pile arithmétique dans le
               ;registre à virgule flottante 260 (résultat de la
               ;différence).
```



MULF

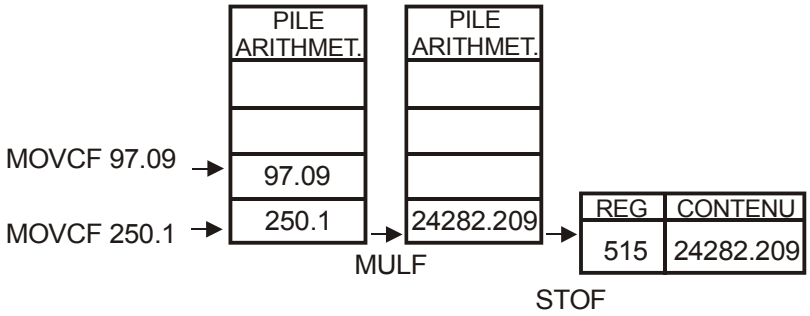
MNÉMONIQUE : **MULF**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **51**

DESCRIPTION :

MULTIPLIE les deux dernières données à virgule flottante de la pile arithmétique entre elles et place le résultat dans la pile arithmétique. Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
MOVCF   250.1 ;Charge la constante 250.1 dans la pile
              ;arithmétique.
MOVCF   97.09 ;Charge la constante 97.09 dans la pile
              ;arithmétique.
MULF          ;Multiplie les deux dernières données contenues
              ;dans la pile arithmétique entre elles et place le
              ;résultat, c'est-à-dire 24282.209, dans cette même
              ;pile.
STOF    515   ;Stocke le contenu de la pile arithmétique dans le
              ;registre à virgule flottante 515 (résultat de la
              ;multiplication).
```



DIVF

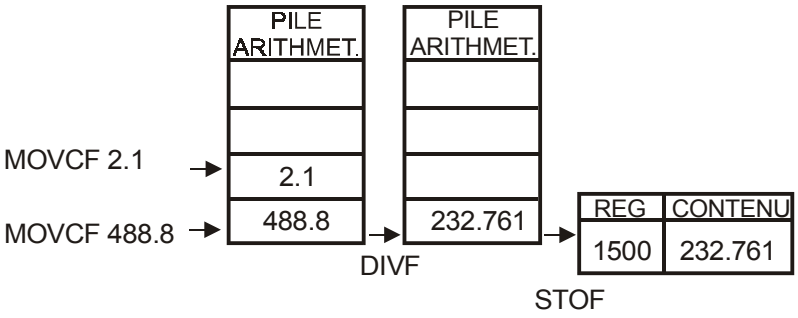
MNÉMONIQUE : **DIVF**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **52**

DESCRIPTION :

DIVISE les deux dernières données entières de la pile arithmétique entre elles et place le résultat dans la pile arithmétique.
Le diviseur est le dernier état de la pile arithmétique. Le dividende est le précédent.
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
MOVCF   488.8 ;Charge la constante 488.8 dans la pile
              ;arithmétique.
MOVCF   2.1   ;Charge la constante 2.1 dans la pile arithmétique.
DIVF          ;Divise les deux dernières données contenues
              ;dans la pile arithmétique entre elles et place le
              ;résultat, c'est-à-dire 232.761, dans cette même
              ;pile.
STOF    1500  ;Stocke le contenu de la pile arithmétique dans le
              ;registre à virgule flottante 1500 (résultat de la
              ;division).
```



MOVIF

MOVIF XXXX

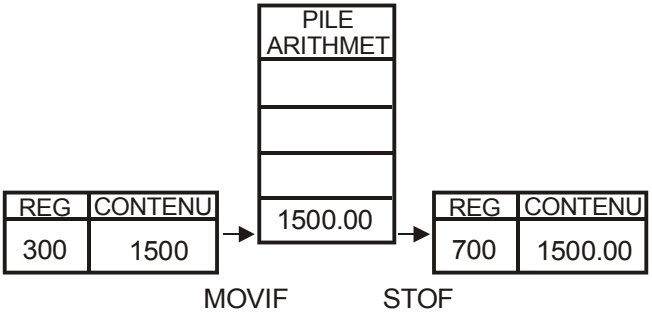
MNÉMONIQUE : **MOVIF**
OPÉRANDE XXXX : **Registre entier de 16 bits**
CODE INSTRUCTION : **25**

DESCRIPTION :

CHARGE le contenu d'un registre entier dans la pile arithmétique et le convertit au format à virgule flottante.
L'opérande est le registre entier dont le contenu est converti au format à virgule flottante, puis placé dans la pile arithmétique. Cela revient à convertir une donnée entière des 16 bits en donnée à virgule flottante de 32 bits.
Incrémente la pile arithmétique de 2 niveaux (+2).

☑ Exemple :

```
MOVIF   300   ;Charge le contenu du registre entier 300 (par
              ;exemple 1005) dans la pile arithmétique et le
              ;convertit au format à virgule flottante (1005.00).
STOF    700   ;Stocke le contenu de la pile arithmétique dans le
              ;registre à virgule flottante 700.
```



STOFI

STOFI XXXX

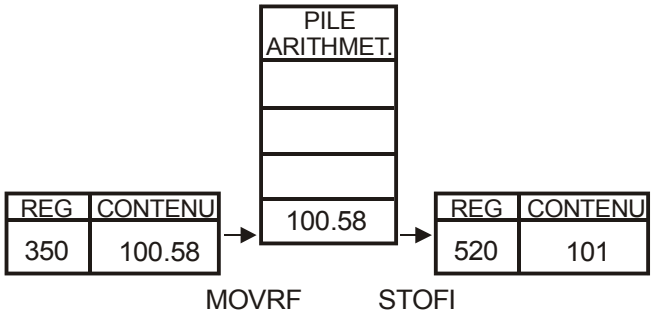
MNÉMONIQUE : **STOFI**
OPÉRANDE XXXX : **Registre de 16 bits à virgule flottante**
CODE INSTRUCTION : **29**

DESCRIPTION :

STOCKE la dernière donnée à virgule flottante de la pile arithmétique dans un registre entier après l'avoir convertie en donnée entière.
L'opérande est l'adresse du registre entier.
La donnée à virgule flottante de la pile arithmétique est arrondie pour être stockée au format entier (elle est arrondie par excès lorsque la décimale est supérieure à 0,5 et par défaut dans le cas contraire).
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
MOVRF   350   ;Charge le contenu du registre à virgule flottante
              ;350, par exemple 100.58, dans la pile
              ;arithmétique.
STOFI   520   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 520 en l'arrondissant au format
              ;entier (101).
```



☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- L'exemple choisi consiste à effectuer l'opération mathématique suivante :

$$(60 + 100) / 2 = R$$

où R est le résultat de l'opération qui figurera sur l'écran.

```
MOVCI    60     ;Charge la constante entière 60 dans la pile.
MOVCI    100    ;Charge la constante entière 100 dans la pile.
ADDI            ;Fait la somme des deux constantes.
MOVCI    2      ;Charge la constante entière 2 dans la pile.
DIVI            ;Divise le résultat de la somme ci-dessus par la
                ;constante 2.
STOI     400    ;Stocke le résultat de l'opération dans le registre
                ;entier 400.
CLEAR           ;Efface la mémoire-tampon intermédiaire et place
                ;le pointeur d'affichage sur la première position.
DISRI    400 3  ;Copie le contenu du registre entier 400 (à trois
                ;chiffres) dans la mémoire-tampon intermédiaire.
COM      0      ;Affiche le contenu de la mémoire-tampon
                ;intermédiaire sur le LCD.
END
```

B.- Cet exemple est similaire au précédent, à une différence près : il s'agit d'afficher les résultats intermédiaires sur trois LCD (exemple exécuté sur un équipement doté de trois écrans).
L'opération mathématique à effectuer est la suivante :

$$[(60 + 100) / 2] * 40 = R$$

Le résultat de la somme s'affichera sur le LCD 1.
Le résultat de la division s'affichera sur le LCD 2.
Le résultat de la multiplication s'affichera sur le LCD 3.

```
MOVCI   60      ;Charge la constante entière 60 dans la pile.
MOVCI   100     ;Charge la constante entière 100 dans la pile.
ADDI            ;Fait la somme des deux constantes.
STOI    400     ;Stocke le résultat de la somme dans le registre
                ;entier 400.
MOVRI   400     ;Charge le contenu du registre entier 400 dans la
                ;pile arithmétique.
MOVCI   2       ;Charge la constante entière 2 dans la pile
                ;arithmétique.
DIVI            ;Divise les données de la pile.
STOI    410     ;Stocke le résultat de la division dans le registre
                ;entier 410.
MOVRI   410     ;Charge le contenu du registre entier 410 dans la
                ;pile arithmétique.
MOVCI   40      ;Charge la constante entière 40 dans la pile
                ;arithmétique.
MULI            ;Multiplie les données de la pile.
STOI    420     ;Stocke le résultat de la multiplication dans le
                ;registre entier 420.
CLEAR           ;Efface la mémoire-tampon intermédiaire et place
                ;le pointeur d'affichage sur la première position.
DISRI   400 3   ;Copie le contenu du registre entier 400 (à trois
                ;chiffres) dans la mémoire-tampon intermédiaire.
COM     0       ;Affiche le contenu de la mémoire-tampon
                ;intermédiaire sur le LCD 1.
CLEAR           ;Efface la mémoire-tampon intermédiaire et place
                ;le pointeur d'affichage sur la première position.
DISRI   410 3   ;Copie le contenu du registre entier 410 (à trois
                ;chiffres) dans la mémoire-tampon intermédiaire.
COM     1       ;Affiche le contenu de la mémoire-tampon
                ;intermédiaire sur le LCD 2.
CLEAR           ;Efface la mémoire-tampon intermédiaire et place
                ;le pointeur d'affichage sur la première position.
DISRI   420 3   ;Copie le contenu du registre entier 420 (à trois
                ;chiffres) dans la mémoire-tampon intermédiaire.
COM     2       ;Affiche le contenu de la mémoire-tampon
                ;intermédiaire sur le LCD 3.
END
```

C.- Cet exemple consiste à écrire les instructions qui permettront d'exécuter la formule suivante (employée pour linéariser une sonde PT100) :

$$T = -244.913 + 0.0234\ (K * ent\_an) + 0.0137\ (K * ent\_an)^2$$

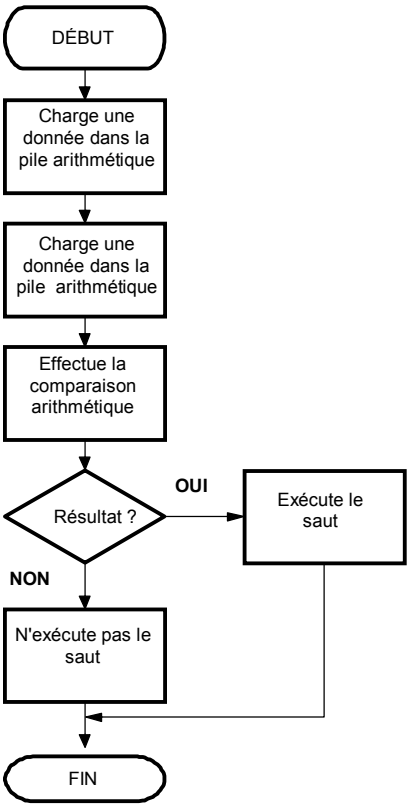
où K est la constante et ent_an l'une des entrées analogiques d'un équipement MIDA.

```
;Définitions des étiquettes
;
ent_an      equ   xxx   ;xxx = registre entier équivalent à une entrée
                        ;analogique.
cons_k      equ   xxx   ;xxx = registre à virgule flottante contenant la
                        ;constante K.
inter       equ   xxx   ;xxx = registre à virgule flottante contenant le
                        ;résultat des opérations.
resultat    equ   xxx   ;xxx = registre à virgule flottante contenant le
                        ;résultat de l'opération.
;
        MOVIF     ent_an      ;Charge le contenu du registre entier
                              ;"ent_an" dans la pile arithmétique en le
                              ;convertissant en donnée à virgule flottante.
        MOVRF     cons_k      ;Charge le contenu du registre à virgule
                              ;flottante "cons_k" dans la pile
                              ;arithmétique.
        MULF                  ;Multiplie les deux dernières données de la
                              ;pile arithmétique entre elles et place le
                              ;résultat dans cette même pile.
        STOF      inter       ;Stocke le contenu de la pile arithmétique
                              ;dans le registre à virgule flottante
                              ;"inter", résultat de la multiplication
                              ;(K * ent_an).
        MOVCF     -244.913    ;Charge la constante -244.913 dans la pile
                              ;arithmétique.
        MOVRF     inter       ;Charge le contenu du registre à virgule
                              ;flottante "inter" dans la pile arithmétique.
        MOVCF     0.0234      ;Charge la constante 0.0234 dans la pile
                              ;arithmétique.
```

```
MULF            ;Multiplie les deux dernières données
                ;contenues dans la pile arithmétique entre
                ;elles et place le résultat dans cette même
                ;pile (0.0234 * "inter").
ADDF            ;Fait la somme des deux dernières
                ;données contenues dans la pile
                ;arithmétique et place le résultat dans cette
                ;même pile (-244.913 + le résultat de la
                ;multiplication précédente).
MOVRF   inter   ;Charge le contenu du registre à virgule
                ;flottante "inter" dans la pile arithmétique.
MOVCF   0.0137  ;Charge la constante 0.0137 dans la pile
                ;arithmétique.
MULF            ;Multiplie les deux dernières données
                ;contenues dans la pile arithmétique entre
                ;elles et place le résultat dans cette même
                ;pile (0.0137 * "inter").
ADDF            ;Fait la somme des deux dernières
                ;données contenues dans la pile
                ;arithmétique et place le résultat dans cette
                ;même pile (le résultat de la somme
                ;précédente + le résultat de la dernière
                ;multiplication).
STOF    resultat ;Stocke le contenu de la pile arithmétique
                ;dans un registre à virgule flottante
                ;(résultat de la somme).
```

L'équipement dispose d'instructions permettant de réaliser toutes sortes de comparaisons arithmétiques entre diverses constantes, entre le contenu de plusieurs registres, et entre diverses constantes et le contenu de registres. Ces instructions permettent de déterminer :

- ***Si deux données sont ÉGALES.***
- ***Si une donnée est SUPÉRIEURE ou ÉGALE à une autre.***
- ***Si une donnée est INFÉRIEURE ou ÉGALE à une autre.***
- ***Si une donnée est SUPÉRIEURE à une autre.***
- ***Si une donnée est INFÉRIEURE à une autre.***



Toutes les comparaisons peuvent être réalisées au format entier ou à virgule flottante, mais les données comparées doivent avoir le même format.
Les comparaisons s'effectuent par l'intermédiaire de la pile arithmétique.

Le diagramme de flux décrit la manière dont les opérations de comparaison sont exécutées. Les instructions ci-après permettent de charger les données dans la pile arithmétique :

- MOVRI ou MOVCI pour charger des données entières (registres ou constantes).

- MOVRF ou MOVCF pour charger des données à virgule flottante (registres ou constantes).

✓ LISTE DES INSTRUCTIONS

Les instructions de ce groupe sont :

CPEI	**CPEF**
CPGEI	**CPGEF**
CPLEI	**CPLEF**
CPGI	**CPGF**
CPLI	**CPLF**

Nous vous les décrivons ci-après en détail :

CPEI

CPEI XXXX

MNÉMONIQUE : **CPEI**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **60**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données entières de la pile arithmétique indique que ces données sont ÉGALES.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut.
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
0000  MOVRI     300    ;Charge le registre entier 300 dans la pile
                         ;arithmétique.
0001  MOVCI     2      ;Charge la constante 2 dans la pile arithmétique.
0002  CPEI      100    ;Si le contenu du registre entier 300 est égal à 2,
                         ;le programme saute à la ligne 100. Dans le cas
                         ;contraire, il passe à la ligne 3.
0003  ---------
0004  ---------
------ ---------
0100  ---------
```

Ce programme vérifie si les données introduites dans la pile arithmétique sont égales. Le programme saute à la ligne 100 si les données comparées sont égales. Dans le cas contraire, il passe à la ligne 3.

CPGEI

CPGEI XXXX

MNÉMONIQUE : **CPGEI**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **61**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données entières de la pile arithmétique indique que la donnée introduite en premier lieu est SUPÉRIEURE ou ÉGALE à la suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut. C'est la dernière donnée de la pile arithmétique qui sert de référence pour la comparaison.
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
0030  MOVRI     350   ;Charge le contenu du registre 350 dans la pile
                      ;arithmétique.
0031  MOVCI     42    ;Charge la constante 42 dans la pile arithmétique.
0032  CPGEI     485   ;Si le contenu du registre 350 est supérieur ou
                      ;égal à 42, le programme saute à la ligne 485.
                      ;Dans le cas contraire, il passe à la ligne 33.
0033  ---------
0034  ---------
------  ---------
0485  ---------
```

Si le contenu du registre introduit en premier lieu dans la pile arithmétique est supérieur ou égal à la constante introduite en second lieu, le programme saute à la ligne 485. Dans le cas contraire, il passe à la ligne 33.

CPLEI

CPLEI XXXX

MNÉMONIQUE : **CPLEI**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **62**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données entières de la pile arithmétique indique que la donnée introduite en premier lieu est INFÉRIEURE ou ÉGALE à la suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut. C'est la dernière donnée de la pile arithmétique qui sert de référence pour la comparaison.
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
0102  MOVRI     421   ;Charge le contenu du registre entier 421 dans la
                      ;pile arithmétique.
0103  MOVCI     -250  ;Charge la constante –250 dans la pile
                      ;arithmétique.
0104  CPLEI     1050  ;Si le contenu du registre entier 421 est inférieur
                      ;ou égal à -250, le programme saute à la ligne
                      ;1050. Dans le cas contraire, il passe à la ligne
                      ;105.
0105  ---------
0106  ---------
------  ---------
1050  ---------
```

Si le contenu du registre introduit en premier lieu dans la pile arithmétique est inférieur ou égal à la constante introduite en second lieu, le programme saute à la ligne 1050. Dans le cas contraire, il passe à la ligne 105.

CPGI

CPGI XXXX

MNÉMONIQUE : **CPGI**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **63**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données entières de la pile arithmétique indique que la donnée introduite en premier lieu est SUPÉRIEURE à la suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut .
C'est la dernière donnée de la pile arithmétique qui sert de référence pour la comparaison.
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
0010  MOVRI     930   ;Charge le contenu du registre entier 930 dans la
                      ;pile arithmétique.
0011  MOVCI     21    ;Charge la constante 21 dans la pile arithmétique.
0012  CPGI      90    ;Si le contenu du registre entier 930 est supérieur
                      ;à 21, le programme saute à la ligne 90. Dans le
                      ;cas contraire, il passe à la ligne 13.
0013  ---------
0014  ---------
------ ---------
0090  ---------
```

Si le contenu du registre introduit en premier lieu dans la pile arithmétique est supérieur à la constante introduite en second lieu, le programme saute à la ligne 90. Dans le cas contraire, il passe à la ligne 13.

CPLI

CPLI XXXX

MNÉMONIQUE : **CPLI**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **64**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données entières de la pile arithmétique indique que la donnée introduite en premier lieu est INFÉRIEURE à la suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut. C'est la dernière donnée de la pile arithmétique qui sert de référence pour la comparaison.
Décrémente la pile arithmétique de 2 niveaux (-2).

☑ Exemple :

```
0035  MOVCI     282   ;Charge la constante 282 dans la pile
                       ;arithmétique.
0036  MOVRI     300   ;Charge le contenu du registre entier 300 dans la
                       ;pile arithmétique.
0037  CPLI      900   ;Si le contenu du registre entier 300 est inférieur
                       ;à 282, le programme saute à la ligne 900. Dans
                       ;le cas contraire, il passe à la ligne 38.
0038  ---------
0039  ---------
------ ---------
0900  ---------
```

Si la constante introduite en premier lieu dans la pile arithmétique est inférieure au contenu du registre entier introduit en second lieu, le programme saute à la ligne 900. Dans le cas contraire, il passe à la ligne 38.

CPEF

CPEF XXXX

MNÉMONIQUE : **CPEF**
OPÉRANDE XXXX : **Numéro de ligne du programme**
CODE INSTRUCTION : **54**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données à virgule flottante de la pile arithmétique indique que ces données sont ÉGALES.
L'opérande est le numéro de la ligne de destination du saut.
Décrémente la pile arithmétique de 4 niveaux (-4).

☑ Exemple :

```
0000  MOVRF     100   ;Charge le contenu du registre à virgule flottante
                       ;100, par exemple 2.10001, dans la pile
                       ;arithmétique.
0001  MOVCF     2.1   ;Charge la constante 2.1 dans la pile arithmétique.
0002  CPEF      120   ;Si le contenu du registre à virgule flottante 100
                       ;est égal à 2.10001, le programme saute à la ligne
                       ;120. Dans le cas contraire, il passe à la ligne 3.
0003  ---------
0004  ---------
------  ---------
0120  ---------
```

Si le contenu du registre à virgule flottante introduit en premier lieu dans la pile arithmétique est égal à la constante introduite en second lieu, le programme saute à la ligne 120. Dans le cas contraire, il passe à la ligne 3. Il convient de signaler que les données comparées doivent être rigoureusement identiques, y compris au niveau des décimales. Pour revenir à l'exemple ci-dessus, si la première donnée introduite est 2.1001 et la seconde 2.1, le programme passe à la ligne 3.

CPGEF

CPGEF XXXX

MNÉMONIQUE : **CPGEF**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **55**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données à virgule flottante de la pile arithmétique indique que la donnée introduite en premier lieu est SUPÉRIEURE ou ÉGALE à la suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut. C'est la dernière donnée de la pile arithmétique qui sert de référence pour la comparaison.
Décrémente la pile arithmétique de 4 niveaux (-4).

☑ Exemple :

```
0002  MOVRF    100   ;Charge le contenu du registre à virgule flottante
                      ;100 dans la pile arithmétique.
0003  MOVCF    2.1   ;Charge la constante 2.1 dans la pile arithmétique.
0004  CPGEF    190   ;Si le contenu du registre à virgule flottante 100
                      ;est supérieur ou égal à 2.1, le programme saute à
                      ;la ligne 190. Dans le cas contraire, il passe à la
                      ;ligne 5.
0005  ---------
------  ---------
0190  ---------
```

Si le contenu du registre à virgule flottante introduit en premier lieu dans la pile arithmétique est supérieur ou égal à la constante introduite en second lieu, le programme saute à la ligne 190. Dans le cas contraire, il passe à la ligne 5.

CPLEF

CPLEF XXXX

MNÉMONIQUE : **CPLEF**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **56**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données à virgule flottante de la pile arithmétique indique que la donnée introduite en premier lieu est INFÉRIEURE ou ÉGALE à la suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut. C'est la dernière donnée de la pile arithmétique qui sert de référence pour la comparaison.
Décrémente la pile arithmétique de 4 niveaux (-4).

☑ Exemple :

```
0000  MOVRF    212    ;Charge le contenu du registre à virgule flottante
                      ;212 dans la pile arithmétique.
0001  MOVCF    6.82   ;Charge la constante 6.82 dans la pile
                      ;arithmétique.
0002  CPLEF    100    ;Si le contenu du registre à virgule flottante 212
                      ;est inférieur ou égal à 6.82, le programme saute à
                      ;la ligne 100. Dans le cas contraire, il passe à la
                      ;ligne 3.
0003  ---------
0004  ---------
------  ---------
0100  ---------
```

Si le contenu du registre à virgule flottante introduit en premier lieu dans la pile arithmétique est inférieur ou égal à la constante introduite en second lieu, le programme saute à la ligne 100. Dans le cas contraire, il passe à la ligne 3.

CPGF

CPGF XXXX

MNÉMONIQUE : **CPGF**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **57**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données à virgule flottante de la pile arithmétique indique que la donnée introduite en premier lieu est SUPÉRIEURE à la suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut. C'est la dernière donnée de la pile arithmétique qui sert de référence pour la comparaison.
Décrémente la pile arithmétique de 4 niveaux (-4).

☑ Exemple :

```
0000  MOVRF     805    ;Charge le contenu du registre à virgule flottante
                        ;805 dans la pile arithmétique.
0001  MOVCF     213.7  ;Charge la constante 213.7 dans la pile
                        ;arithmétique.
0002  CPGF      1000   ;Si le contenu du registre à virgule flottante 805
                        ;est supérieur à 213.7, le programme saute à la
                        ;ligne 1000. Dans le cas contraire, il passe à la
                        ;ligne 3.
0003  ---------
0004  ---------
------  ---------
1000  ---------
```

Si le contenu du registre à virgule flottante introduit en premier lieu dans la pile arithmétique est supérieur à la constante introduite en second lieu, le programme saute à la ligne 1000. Dans le cas contraire, il passe à la ligne 3.

CPLF

CPLF XXXX

MNÉMONIQUE : **CPLF**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **58**

DESCRIPTION :

Le programme effectue un SAUT si la comparaison entre les deux dernières données à virgule flottante de la pile arithmétique indique que la donnée introduite en premier lieu est INFÉRIEURE à la suivante.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut. C'est la dernière donnée de la pile arithmétique qui sert de référence pour la comparaison.
Décrémente la pile arithmétique de 4 niveaux (-4).

☑ Exemple :

```
0000  MOVCF     1.99   ;Charge la constante 1.99 dans la pile
                        ;arithmétique.
0001  MOVRF     210    ;Charge le contenu du registre à virgule flottante
                        ;210 dans la pile arithmétique.
0002  CPLF      100    ;Si la constante est inférieure au contenu du
                        ;registre à virgule flottante 210, le programme
                        ;saute à la ligne 100. Dans le cas contraire, il
                        ;passe à la ligne 3.
0003  ---------
0004  ---------
------  ---------
0100  ---------
```

Si la constante introduite en premier lieu dans la pile arithmétique est inférieure au contenu du registre à virgule flottante introduit en second lieu, le programme saute à la ligne 100. Dans le cas contraire, il passe à la ligne 3.

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Programmation d'un contrôleur de température simple qui déclenche une alarme et coupe les dispositifs de chauffage lorsque la température dépasse 70ºC et met ces dispositifs en marche lorsque la température est inférieure à 70ºC. Il s'agit donc de commandes de type TOUT OU RIEN.
Le programme doit donc contrôler une première sortie numérique en fonction d'une valeur de température et une seconde sortie qui déclenche une alarme lorsque la température atteint ou dépasse 70ºC.

```
;Définition des libellés
texte0      lite    “AL.TEMP”
;
debut MOVRF     400         ;Charge le contenu du registre à virgule
                            ;flottante dans la pile arithmétique (le
                            ;contenu du registre est la température,
                            ;convertie en degrés et traitée dans une
                            ;autre zone du programme).
      STOF      401         ;Stocke le contenu de la pile arithmétique
                            ;dans le registre à virgule flottante 401
                            ;(registre de température).
      MOVRF     401         ;Charge le contenu du registre à virgule
                            ;flottante dans la pile arithmétique.
      MOVCF     70          ;Charge la constante 70 à virgule flottante
                            ;dans la pile arithmétique.
      CPGEF     alar        ;Compare le contenu du registre à virgule
                            ;flottante 401 et la constante 70. Si le
                            ;contenu de ce registre est supérieur ou
                            ;égal à la constante 70, le programme
                            ;saute à la ligne "alar".
      CLEAR                 ;Efface la mémoire-tampon intermédiaire
                            ;et place le pointeur d'affichage en
                            ;première position.
      DISRF     401   62    ;Copie le contenu du registre à virgule
                            ;flottante 401 dans la mémoire-tampon
                            ;intermédiaire.
```

```
        COM     0       ;Affiche le contenu de la mémoire-tampon
                        ;intermédiaire sur le LCD.
        SET     105     ;Active le relais 105 (mise en marche des
                        ;dispositifs de chauffage).
        RESET   106     ;Désactive le relais 106 (alarme
                        ;température excessive).
        JNZ     debut
alar    RESET   105     ;Désactive le relais 105 (mise en marche
                        ;des dispositifs de chauffage).
        SET     106     ;Active le relais 106 (alarme température
                        ;excessive).
        CLEAR           ;Efface la mémoire-tampon intermédiaire
                        ;et place le pointeur d'affichage en
                        ;première position.
        DISL    texte0  ;Charge le message "texte0" de la table de
                        ;libellés dans la mémoire-tampon
                        ;intermédiaire.
        COM     0       ;Affiche le contenu de la mémoire-tampon
                        ;intermédiaire sur le LCD.
        END
```

Si vous souhaitez vous rendre compte immédiatement de la manière dont ce programme fonctionne, vous pouvez remplacer l'instruction MOVRF 400 par MOVCF 80. Le message AL.TEMP s'affichera sur le LCD. Si au lieu de remplacer l'instruction MOVRF 400 par MOVCF 80, vous la remplacez par MOVCF 60 le contenu du registre à virgule flottante 401 s'affichera.

Les équipements MIDA sont dotés d'un clavier intégré. Ce clavier peut être utilisé pour activer ou désactiver des sorties ou des relais internes, introduire des données numériques, mettre l'horloge interne de l'équipement à l'heure et créer des menus personnalisés.

Les ***instructions liées*** au clavier, qui dépendent de l'équipement MIDA dont vous disposez, sont les suivantes :

INK	Détecte l'activation d'une touche donnée.
INI	Permet d'introduire des données entières.
INF	Permet d'introduire des données à virgule flottante.
INPIX	Permet d'introduire des données entières, sans interrompre le programme.
INPFX	Permet d'introduire des données à virgule flottante, sans interrompre le programme.
INICF	Permet d'introduire des données entières.
INPCX	Permet d'introduire des données entières, sans interrompre le programme.

L'instruction **"LD"** peut également être classée dans ce groupe. Elle permet de détecter l'activation d'une touche donnée.

Les actions pouvant être réalisées au moyen des instructions ci-dessus sont les suivantes :

- ***Détection de l'activation d'une touche.***
- ***Introduction de données entières ou à virgule flottante.***
- ***Création de menus personnalisés.***

✓ DÉTECTION DE TOUCHES

L'activation de toute touche du clavier peut être détectée au moyen de l'instruction **"INK"** ou en consultant l'état du relais interne associée à la touche en question à l'aide de l'instruction **"LD"**.

Détection de l'activation des touches

Toutes les touches se voient assigner un code qui coïncide avec le relais associé à chacune d'entre elles (voir Adressage de la mémoire dans le Manuel Utilisateur de l'équipement MIDA).

L'équipement possède un indicateur interne d'activation de touche. L'instruction **INK** transfère un "1" à la pile logique si cet indicateur *est sur "1"* et le code de son opérande coïncide avec celui de la dernière touche qui a été activée.

Cette instruction transfère un "0" à la pile logique si ces codes ne coïncident pas ou si aucune touche n'a été activée.

L'indicateur interne d'activation de touche est effacé chaque fois que le résultat d'une instruction **INK** est "1".

La détection se répète automatiquement (comme celle des touches d'un PC). Toute touche maintenue enfoncée est détectée toutes les 0,5 secondes, après une attente initiale de 2 secondes.

Le résultat de la détection est uniquement "1" pendant l'exploration d'un programme.

Détection de l'état des touches

Des relais sont associés à chacune des touches (voir Adressage de la mémoire dans le Manuel Utilisateur de chaque équipement). Ces relais sont le reflet de l'état de chacune des touches.

L'instruction **LD** permet de détecter l'activation des touches comme s'il s'agissait de boutons externes. Exemple :

```
LD    30    ;Détection de la touche <ENTER> d'un
            ;équipement MIDA 14.
OUT   100   ;La sortie numérique 100 reste activée tant que la
            ;touche <ENTER> reste enfoncée.
```

Il est important de respecter les instructions de programmation décrites dans le chapitre Instructions logiques et de saut de ce Manuel technique.

INK

INK XXXX

MNÉMONIQUE : **INK**
OPÉRANDE XXXX : **Constante indiquant le code de la touche**
CODE INSTRUCTION : **85**

DESCRIPTION :

CHARGE un état "1" dans la pile logique si le code de la dernière touche activée coïncide avec celui désigné par l'opérande. Dans le cas contraire, l'état chargé est "0".
L'opérande XXXX est le code de la touche qui doit être détectée.
Le code de la touche diffère selon les équipements (consultez le Manuel Utilisateur de l'équipement MIDA correspondant).
La dernière touche activée figure dans un registre qui s'active lorsque vous appuyez sur n'importe quelle touche et se désactive lorsque l'instruction INK reconnaît la touche recherchée.
Incrémente la pile logique de 1 niveau (+1).

☑ Exemple :

```
INK   340    ;Détection de la touche <ENTER>d'un
             ;équipement MIDA 64.
JNZ   100    ;Si la dernière touche ayant été activée est
             ;<ENTER>, le programme saute à la ligne 100.
INK   344    ;Détection de la touche <7>.
JNZ   200    ;Si la dernière touche ayant été activée est <7>, le
             :programme passe à la ligne 200.
INK   341    ;Détection de la touche <CLEAR>.
OUT   500    ;Lorsque vous appuyez sur la touche <CLEAR>,
             ;le relais interne 500 s'active pendant une
             ;exploration de programme.
```

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Déchargement de l'activation d'une touche sur un relais interne quelconque :

```
INK  63   ;Détection du relais correspondant à une touche.
OUT  500  ;Le relais interne 500 est "1" chaque fois que le
          ;programme détecte l'activation de cette touche.
```

Le résultat de la détection est uniquement "1" pendant une exploration de programme.

B.- Arrêt/Marche au moyen de deux touches données.

```
INK     340  ;Détection du relais correspondant à une touche
             ;(arrêt).
OUT     400  ;Décharge l'activation de la touche pendant une
             ;exploration de programme.
INK     344  ;Détection du relais correspondant à une touche
             ;(marche).
OUT     401  ;Décharge l'activation de la touche pendant une
             ;exploration de programme.
LD      400
OR      101
ANDNT   401
OUT     101  ;Réalisation d'une maille d'arrêt/de marche simple.
```

C.- Élaboration d'un menu comportant trois options et répondant à l'activation des touches <ENTER>, <UP> et <CLEAR> pour dévier le flux de programme vers trois sous-routines différentes dans lesquelles elles activeront des relais.

```
;Table des libellés
texte0    lite   “PREMIER MENU”
texte1    lite   “DEUXIÈME MENU”
texte2    lite   “TROISIÈME MENU”
```

```
;
touc0 INK       60      ;Détection de la touche <ENTER> (par exemple).
      JZ        touc1   ;Si la touche <ENTER> a été activée, le
                        ;programme passe à la ligne suivante. Dans le
                        ;cas contraire, il saute à l'étiquette de programme
                        ;"touc1".
      CLEAR             ;Efface la mémoire-tampon intermédiaire et place
                        ;le pointeur d'affichage sur la première position.
      DISL      0       ;Copie le premier message de la table des libellés
                        ;dans la mémoire-tampon intermédiaire.
      COM       0       ;Affiche le contenu de la mémoire-tampon
                        ;intermédiaire sur le LCD.
      CALL      rout1   ;Appel de la routine de programme "rout1".
touc1 INK       62      ;Détection de la touche <CLEAR> (par exemple).
      JZ        touc2   ;Si la touche <CLEAR> a été activée, le
                        ;programme passe à la ligne suivante. Dans le
                        ;cas contraire, il saute à l'étiquette de programme
                        ;"touc2".
      CLEAR             ;Efface la mémoire-tampon intermédiaire et place
                        ;le pointeur d'affichage sur la première position.
      DISL      1       ;Copie le second message de la table des libellés
                        ;dans la mémoire-tampon intermédiaire.
      COM       0       ;Affiche le contenu de la mémoire-tampon
                        ;intermédiaire sur le LCD.
      CALL      rout2   ;Appel de la routine de programme "rout2".
touc2 INK       63      ;Détection de la touche <UP> (par exemple).
      JZ        touc1   ;Si la touche <UP> a été activée, le
                        ;programme passe à la ligne suivante. Dans le
                        ;cas contraire, il saute à l'étiquette de programme
                        ;"touc1".
      CLEAR             ;Efface la mémoire-tampon intermédiaire et place
                        ;le pointeur d'affichage sur la première position.
      DISL      2       ;Copie le troisième message de la table des
                        ;libellés dans la mémoire-tampon intermédiaire.
      COM       0       ;Affiche le contenu de la mémoire-tampon
                        ;intermédiaire sur le LCD.
      CALL      rout3   ;Appel de la routine de programme "rout3".
      END
rout1 RESET     101     ;Désactive le relais 101.
      RESET     102     ;Désactive le relais 102.
      SET       100     ;Active le relais 100.
```

```
        RET                 ;Passe à la ligne suivante de programme depuis
                            ;laquelle l'appel (CALL) de cette sous-routine a eu
                            ;lieu.
rout2   RESET       100     ;Désactive le relais 100.
        RESET       102     ;Désactive le relais 102.
        SET         101     ;Active le relais 101.
        RET                 ;Passe à la ligne suivante de programme depuis
                            ;laquelle l'appel (CALL) de cette sous-routine a eu
                            ;lieu.
rout3   RESET       100     ;Désactive le relais 100.
        RESET       101     ;Désactive le relais 101.
        SET         102     ;Active le relais 102.
        RET                 ;Passe à la ligne suivante de programme depuis
                            ;laquelle l'appel (CALL) de cette sous-routine a eu
                            ;lieu.
        END
```

✓ INTRODUCTION DE DONNÉES NUMÉRIQUES

Il existe deux groupes d'instructions permettant l'introduction de données numériques :

- avec interruption du programme utilisateur.
- sans interruption du programme utilisateur.

Introduction de données avec interruption du programme

Les instructions **INI, INF** et **INICF** permettent, selon le cas, d'introduire des données numériques entières ou à virgule flottante par l'intermédiaire du clavier de l'équipement (codes d'accès, présélection de valeur de temporisation ou de comptage, consignes d'alarme, etc.).

Lorsque le programme exécute l'une de ces instructions, quelle qu'elle soit, un certain nombre d'astérisques s'affichent dans la zone du LCD définie dans le programme. Le nombre d'astérisques affichés dépend du format fixé dans l'opérande de l'instruction. L'introduction des données numériques s'effectue au moyen des touches de l'équipement. En cas d'erreur lors de l'introduction d'une donnée, celle-ci peut être effacée grâce à la touche <CLEAR>.

La phase d'introduction s'achève lorsque vous appuyez sur la touche <ENTER> pour confirmer la donnée introduite. L'équipement attend donc l'activation de la touche <ENTER> pour poursuivre l'exécution du programme. N'oubliez donc pas, lorsque utilisez ces instructions, que l'exécution du programme s'interrompt jusqu'à ce que la donnée entrée soit validée.

REMARQUE : *Ces instructions **interrompent** l'exécution du programme utilisateur.*

Comme nous l'avons déjà mentionné plus haut, lorsqu'une vous maintenez une touche enfoncée pendant un certain temps, elle est détectée de manière répétitive toutes les 0,5 secondes, après une attente initiale de 2 secondes.

Les instructions **INI, INF** et **INICF** disposent d'un ou de deux opérandes, et leur comportement diffère selon l'équipement (pour déterminer quelle instruction vous devez utilisez, consultez le Manuel Utilisateur de l'équipement correspondant) :

INI A	(introduction de données entières)
A	Format d'introduction de la donnée ou numéro de LCD sur lequel l'introduction aura lieu (équipements dotés de plus d'un LCD).
INF A B	(introduction de données à virgule flottante)
A	Format d'introduction de la donnée ou numéro de LCD sur lequel l'introduction aura lieu (équipements dotés de plus d'un LCD).
B	Nombre de décimales (équipements dotés de plus d'un LCD).
INICF AB	(introduction de données entières formatées)
AB	Format d'introduction de la donnée.

La donnée numérique entière introduite doit être comprise entre -32768 et 32767 (données entières de 16 bits avec signe).

La position du premier caractère est déterminée par celle du pointeur à l'intérieur de la mémoire-tampon intermédiaire (voir instruction LOC).

Si vous utilisez une instruction INPIX, INPFX, INPCX ou CLKP, l'exécution de celle-ci devra être achevée avant l'exécution d'une instruction INI, INF ou INICF.

Introduction de données sans interruption du programme.

Les instructions **INPIX**, **INPFX** et **INPCX** permettent également d'introduire des données entières ou à virgule flottante, mais contrairement aux instructions précédentes (INI, INF et INICF), elles n'interrompent pas l'exécution du programme utilisateur.

Les instructions **INPIX**, **INPFX** et **INPCX** sont liées aux interruptions. C'est la raison pour laquelle elles n'interrompent pas l'exécution du programme utilisateur. Le microprocesseur de l'équipement MIDA doit par contre traiter un plus grand nombre d'interruptions, ce qui ralentit l'exécution du programme utilisateur.

Dans certains cas, toutes les interruptions ne peuvent pas être traitées simultanément et certaines d'entre elles, les moins prioritaires, se perdent. Il est donc possible que vous rencontriez des problèmes concernant certains messages de communication via RS pendant l'exécution de ces instructions. Ce dysfonctionnement est sporadique et il vous suffit de réitérer vos tentatives pour résoudre ce problème.

Étant donné leur compatibilité avec les instructions DISIX et DISFX dans le cas des formats négatifs (justification à gauche), les instructions **INPIX**, **INPFX** et **INPCX** considèrent la valeur absolue du format pour pouvoir utiliser les mêmes pointeurs que pour les instructions d'affichage (cette compatibilité dépend du modèle d'équipement MIDA que vous êtes en train de programmer).

L'instruction **INPFX** ne tient pas compte du nombre de décimales (mais il est nécessaire de le préciser pour conserver la compatibilité avec l'instruction DISFX), car c'est l'utilisateur lui-même qui fixe la décimale lorsqu'il introduit la donnée. *Exemple :* vous pouvez utiliser le contenu 62 d'un registre entier pour l'instruction **DISFX** (6 caractères, dont deux décimales) et pour l'instruction **INPFX**, mais cette dernière ignorera le format du nombre de décimales.

INPIX A BC (introduction de données entières)	
A	Registre entier contenant l'adresse du registre entier dans lequel la donnée introduite doit être stockée.
BC	Registre entier contenant le format d'introduction de la donnée ou numéro de LCD sur lequel l'introduction aura lieu (équipements dotés de plus d'un LCD).
INPFX A BC (introduction de données à virgule flottante)	
A	Registre entier contenant l'adresse du registre à virgule flottante dans lequel la donnée introduite doit être stockée.
BC	Registre entier contenant le format d'introduction de la donnée ou numéro de LCD sur lequel l'introduction aura lieu, et nombre de décimales (équipements dotés de plus d'un LCD).
INPCX A BC (introduction de données entières formatées)	
A	Registre entier contenant l'adresse du registre à virgule flottante dans lequel la donnée introduite doit être stockée.
BC	Registre entier contenant le format d'introduction de la donnée.

La valeur maximale à virgule flottante (IEEE de 32 bits) pouvant être introduite est celle qui résulte de la limitation du nombre de caractères pouvant être affichés sur les différents écrans de chaque équipement. Les données introduites sont placées dans la pile arithmétique.

La position du premier caractère est déterminée par celle du pointeur à l'intérieur de la mémoire-tampon intermédiaire (voir instruction LOC).

Il est impossible d'exécuter une seconde fois une instruction INPIX, INPFX, INPCX ou CLKP avant que la première exécution de celle-ci soit achevée.

Le contenu du LCD des équipements MIDA qui disposent d'un écran de ce type peut être modifié pendant l'introduction de la donnée. La zone de l'écran non occupée par l'introduction fonctionne normalement. Après confirmation de l'entrée de la donnée, l'écran récupère le contenu envoyé sur la zone occupée.

Si l'équipement MIDA que vous utilisez possède plus d'un LCD, les écrans non affectés par l'instruction fonctionneront normalement. Après confirmation de l'entrée de la donnée, l'écran récupérera le dernier contenu envoyé.

Dans le cas des instructions INPIX, INPFX et INPCX, le contenu stocké dans les registres désignés par les opérandes doit se trouver à l'intérieur des limites fixées.

Si l'adresse de destination du résultat introduite dans le registre entier se trouve en dehors de ces limites, l'erreur correspondante du registre des erreurs de l'équipement s'active (consultez le Manuel Utilisateur de l'équipement). Cette erreur s'active immédiatement après l'exécution de l'instruction.

Si le contenu du registre entier dans lequel se trouve le format de la donnée à introduire est incorrect, il est remplacé par une valeur par défaut.

Un relais interne, "**Relais entrée**" (consultez le Manuel Utilisateur de l'équipement) s'active en cas d'appel de l'une des instructions suivantes : INPIX, INPFX ou INPCX. Il reste activé pendant l'introduction de la donnée et se désactive lorsque vous validez celle-ci au moyen de la touche <ENTER>.

REMARQUE : Le registre de destination de la donnée introduite (au moyen des instructions INI, INF, INICF, INPIX, INPFX ou INPCX) peut uniquement être un registre de la RAM ou de la NOVRAM, ***en aucun cas un registre de l'EEPROM***.

Vous trouverez, ci-après, la description des instructions de chaque groupe :

INI

INI XXXX

MNÉMONIQUE : **INI**
OPÉRANDE XXXX : **Constante indiquant le format d'introduction de la donnée** ou **numéro de LCD dans lequel l'introduction aura lieu.**
CODE INSTRUCTION : **86**

DESCRIPTION :

INTRODUCTION de données numériques entières par l'intermédiaire du clavier de l'équipement.
L'opérande détermine le nombre de caractères (signe compris) dont sera constitué le format d'introduction ou le LCD dans lequel l'introduction aura lieu (équipements MIDA dotés de plus d'un LCD).
L'exécution de cette instruction s'achève par l'activation de la touche <ENTER>.
La donnée introduite est chargée sur la pile arithmétique.
Incrémente la pile arithmétique de 1 niveau (+1).
Interrompt l'exécution du programme.

☑ Exemple pour un équipement MIDA avec LCD à 32 caractères :

```
CLEAR          ;Efface la mémoire-tampon intermédiaire et place
               ;le pointeur d'affichage sur la première position.
DISL    19     ;Copie, dans la mémoire-tampon intermédiaire, le
               ;message situé en vingtième position dans la
               ;table de libellés du programme. Ce message se
               ;réfère à la donnée qui va être introduite.
LOC     20     ;Place le pointeur d'affichage sur la position 21
               ;(centre de la seconde ligne du LCD).
COM     0      ;Copie le contenu de la mémoire-tampon
               ;intermédiaire dans le LCD pour que le message
               ;se référant à la donnée qui va être introduite
               ;s'affiche.
INI     4      ;Permet d'introduire une donnée entière à quatre
               ;caractères.
```

```
STOI    300   ;Stocke, dans le registre entier 300, la donnée
              ;introduite par l'intermédiaire du clavier.
```

☑ Exemple pour un équipement MIDA doté de trois LCD à 4 caractères 1/2 :

```
CLEAR         ;Efface la mémoire-tampon intermédiaire et place
              ;le pointeur d'affichage sur la première position.
DISL    0     ;Copie, dans la mémoire-tampon intermédiaire, le
              ;message situé sur la première position dans la
              ;table de libellés du programme. Ce message se
              ;réfère à la donnée qui va être introduite.
COM     0     ;Copie le contenu de la mémoire-tampon
              ;intermédiaire dans le premier LCD (supérieur)
              ;pour que le message se référant à la donnée qui
              ;va être introduite s'affiche.
CLEAR         ;Efface la mémoire-tampon intermédiaire et place
              ;le pointeur d'affichage sur la première position.
INI     2     ;Permet d'introduire une donnée entière dans le
              ;troisième LCD (inférieur).
STOI    450   ;Stocke, dans le registre entier 450, la donnée
              ;introduite par l'intermédiaire du clavier.
```

INF

INF XXXX YYYY

MNÉMONIQUE : **INF**
OPÉRANDE XXXX : **Constante indiquant le format d'introduction de la donnée** ou **numéro de LCD dans lequel l'introduction aura lieu.**
OPÉRANDE YYYY : **Constante indiquant le nombre de décimales (uniquement dans le cas des équipements MIDA dotés de plus d'un LCD).**
CODE INSTRUCTION : **87**

DESCRIPTION :

INTRODUCTION du numéro de LCD ou de données numériques à virgule flottante par l'intermédiaire du clavier de l'équipement.
L'opérande XXXX détermine le nombre de caractères (signe et séparation de décimale compris) dont sera constitué le format d'introduction ou le LCD dans lequel l'introduction aura lieu (selon l'équipement MIDA).
L'opérande YYYY indique le nombre de décimales (uniquement dans le cas des équipements MIDA dotés de plus d'un LCD).
L'exécution de cette instruction s'achève par l'activation de la touche <ENTER>.
La donnée introduite est chargée sur la pile arithmétique.
Incrémente la pile arithmétique de 2 niveaux (+2).
Interrompt l'exécution du programme.

☑ Exemple pour un équipement MIDA avec LCD à 32 caractères :

```
CLEAR          ;Efface la mémoire-tampon intermédiaire et place
               ;le pointeur sur la première position.
DISL    5      ;Copie, dans la mémoire-tampon intermédiaire, le
               ;message situé en sixième position dans la
               ;table de libellés du programme.
LOC     1      ;Place le pointeur d'affichage sur la position 2 du
               ;LCD.
```

```
COM     0     ;Copie le contenu de la mémoire-tampon
              ;intermédiaire dans le LCD pour que le message
              ;se référant à la donnée qui va être introduite
              ;s'affiche.
INF     8     ;Permet d'introduire une donnée à virgule flottante
              ;à huit caractères.
STOF    300   ;Stocke, dans le registre à virgule flottante 300, la
              ;donnée introduite par l'intermédiaire du clavier.
```

☑ Exemple pour un équipement MIDA doté de trois LCD à 4 caractères 1/2 :

```
CLEAR         ;Efface la mémoire-tampon intermédiaire et place
              ;le pointeur sur la première position.
DISL    1     ;Copie, dans la mémoire-tampon intermédiaire, le
              ;message déclaré dans la position 2 de la table des
              ;libellés.
COM     1     ;Copie le contenu de la mémoire-tampon
              ;intermédiaire dans le deuxième LCD (milieu) pour
              ;que le message se référant à la donnée qui va
              ;être introduite s'affiche.
INF     2 1   ;Permet d'introduire une donnée à virgule flottante
              ;avec une décimale dans le troisième LCD
              ;(inférieur).
STOF    300   ;Stocke, dans le registre à virgule flottante 300, la
              ;donnée introduite par l'intermédiaire du clavier.
```

INICF

INICF XXXX

MNÉMONIQUE : **INICF**
OPÉRANDE XXXX : **Constante indiquant le format d'introduction de la donnée.**
CODE INSTRUCTION : **86**

DESCRIPTION :

INTRODUCTION de données numériques entières par l'intermédiaire du clavier de l'équipement.
L'opérande XXXX détermine le nombre de caractères (signe et séparation de décimale non compris) dont sera constitué le format d'introduction et indique la position de la décimale.
Le format est : AB
où A est le nombre de caractères pouvant être introduits et où B indique la position de la décimale.
Cette instruction détermine le nombre de caractères et de décimales des données introduites par l'intermédiaire du clavier, mais la mise en mémoire s'effectue toujours sans décimales.
Il est donc impossible de stocker un chiffre supérieur à 3.2767 ou inférieur à -327.68, car les données stockées sont respectivement 32767 et - 32767 (donnée entière de 16 bits avec signe).
L'exécution de cette instruction s'achève par l'activation de la touche <ENTER>.
La donnée introduite est chargée sur la pile arithmétique.
Incrémente la pile arithmétique de 1 niveau (+1).
Interrompt l'exécution du programme.

☑ Exemple pour un équipement MIDA avec LCD à 8 caractères :

```
CLEAR       ;Efface la mémoire-tampon intermédiaire et place
            ;le pointeur sur la première position.
LOC     2   ;Place le pointeur d'affichage sur la position 3 du
            ;LCD.
```

```
INICF    32    ;Permet d'introduire une donnée entière à 3
               ;caractères et la séparation de décimale après le
               ;premier caractère.
STOI     400   ;Stocke la donnée introduite par l'intermédiaire du
               ;clavier dans le registre entier 400.
```

INPIX

INPIX XXXX YYYY

MNÉMONIQUE : **INPIX**
OPÉRANDE XXXX : **Registre entier**
OPÉRANDE YYYY : **Registre entier**
CODE INSTRUCTION : **88**

DESCRIPTION :

INTRODUCTION de données numériques entières par l'intermédiaire du clavier de l'équipement.
L'opérande XXXX est le registre entier contenant l'adresse du registre entier dans lequel la donnée introduite doit être stockée.
L'opérande YYYY est le registre entier contenant le format d'introduction de la donnée ou le LCD dans lequel l'introduction aura lieu (équipements MIDA dotés de plus d'un LCD).
Le format est le nombre de caractères à afficher, signe compris. Si le format est négatif, il est justifié à gauche. Dans le cas contraire, il est justifié à droite. Un relais interne, le "Relais entrée" (consultez le Manuel Utilisateur de l'équipement) s'active lorsque vous appelez cette instruction. Il reste activé pendant l'introduction de la donnée et se désactive lorsque vous validez celle-ci au moyen de la touche <ENTER>.
Ne fait pas intervenir les piles (0).
N'interrompt pas l'exécution du programme.

☑ Exemple pour un équipement avec LCD :

```
SETRI   725   400   ;Stocke la donnée 400 dans le registre
                    ;entier 725.
SETRI   300   4     ;Stocke la donnée 4 dans le registre
                    ;entier 300.
INPIX   725   300   ;Permet d'introduire une donnée dont le
                    ;format est le contenu du registre entier
                    ;300 et stocke celle-ci dans le registre
                    ;entier 400, car c'est celui désigné par le
                    ;registre entier 725.
```

INPFX

INPFX XXXX YYYY

MNÉMONIQUE : **INPFX**
OPÉRANDE XXXX : **Registre entier**
OPÉRANDE YYYY : **Registre entier**
CODE INSTRUCTION : **89**

DESCRIPTION :

INTRODUCTION de données numériques entières par l'intermédiaire du clavier de l'équipement.
L'opérande XXXX est le registre entier contenant l'adresse du registre entier dans lequel la donnée introduite doit être stockée.
L'opérande YYYY est le registre entier contenant le format d'introduction de la donnée. Ce format est : ABC
où AB est le nombre de caractères pouvant être introduits ou le LCD dans lequel l'introduction aura lieu (équipements MIDA dotés de plus d'un LCD) et où C est le nombre de décimales (voir compatibilité de format avec l'instruction DISFX au début de ce chapitre). Un relais interne, le "Relais entrée" (consultez le Manuel Utilisateur de l'équipement) s'active lorsque vous appelez cette instruction. Il reste activé pendant l'introduction de la donnée et se désactive lorsque vous validez celle-ci au moyen de la touche <ENTER>.
Ne fait pas intervenir les piles (0).
N'interrompt pas l'exécution du programme.

☑ Exemple pour un équipement doté de plus d'un LCD :

```
SETRI   310  250   ;Stocke la donnée 250 dans le registre
                   ;entier 310.
MOVCI   23         ;Charge la constante 60 dans la pile
                   ;arithmétique.
STOI    251        ;Stocke le contenu de la pile arithmétique
                   ;dans le registre entier 251.
INPFX   310  251   ;Permet d'introduire une donnée dont le
                   ;format est le contenu du registre 251 (6 =
                   ;6 caractères avec signe) et stocke la
                   ;donnée introduite dans le registre à
                   ;virgule flottante 250, car c'est celui
                   ;désigné par le registre entier 310.
```

INPCX

INPCX XXXX YYYY

MNÉMONIQUE : **INPCX**
OPÉRANDE XXXX : **Registre entier**
OPÉRANDE YYYY : **Registre entier**
CODE INSTRUCTION : **88**

DESCRIPTION :

INTRODUCTION de données numériques entières par l'intermédiaire du clavier de l'équipement.
L'opérande XXXX est le registre entier contenant l'adresse du registre entier dans lequel la donnée introduite doit être stockée.
L'opérande YYYY est le registre entier contenant le nombre de caractères (signe et séparation de décimale non compris) dont sera constitué le format d'introduction et indique la position de la décimale.
Le format est : AB
où A est le nombre de caractères pouvant être introduits et où B indique la position de la décimale.
Cette instruction détermine le nombre de caractères et de décimales des données introduites par l'intermédiaire du clavier, mais la mise en mémoire s'effectue toujours sans décimales.
Il est donc impossible de stocker un chiffre supérieur à 3.2767 ou inférieur à -327.68 car les données stockées sont respectivement 32767 et - 32767 (donnée entière de 16 bits avec signe).
L'exécution de cette instruction s'achève par l'activation de la touche <ENTER>.
Un relais interne, le "Relais entrée" (consultez le Manuel Utilisateur de l'équipement) s'active lorsque vous appelez cette instruction. Il reste activé pendant l'introduction de la donnée et se désactive lorsque vous validez celle-ci au moyen de la touche <ENTER>.
La donnée introduite est chargée sur la pile arithmétique.
Incrémente la pile arithmétique de 1 niveau (+1).
N'interrompt pas l'exécution du programme.

☑ Exemple pour un équipement MIDA avec LCD à 8 caractères :

```
MOVCI   500          ;Charge la constante 500 dans la pile
                     ;arithmétique.
STOI    650          ;Stocke le contenu de la pile arithmétique
                     ;dans le registre entier 650.
MOVCI   52           ;Charge la constante 52 dans la pile
                     ;arithmétique.
STOI    700          ;Stocke le contenu de la pile arithmétique
                     ;dans le registre entier 700.
INPCX   650  700     ;Permet d'introduire une donnée dont le
                     ;format est le contenu du registre 700 (5
                     ;caractères et deux décimales) et stocke la
                     ;donnée introduite dans le registre entier
                     ;500, car c'est celui désigné par le registre
                     ;entier 650.
```

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Pour un LCD à 16x2 caractères
Introduction d'une donnée entière à deux caractères avec signe à partir du premier point de la seconde ligne de l'écran. La première ligne du LCD doit être occupée par un texte.
La donnée sera ensuite stockée dans le registre entier 800.

```
CLEAR           ;Efface la mémoire-tampon intermédiaire et
                ;place le pointeur sur la première position.
DISL    texte   ;Copie, dans la mémoire-tampon
                ;intermédiaire, le message de la table de
                ;libellés désigné.
COM     0       ;Affiche le contenu de la mémoire-tampon
                ;intermédiaire sur le LCD.
INI     3       ;Permet d'introduire trois caractères (dont
                ;un signe). Une fois introduits et après
                ;activation de la touche <ENTER>, le
                ;résultat est transféré dans la pile
                ;arithmétique.
STOI    800     ;Stocke le contenu de la pile arithmétique
                ;dans le registre entier 800.
```

Le message suivant s'affiche sur le LCD :

M	E	S	S	A	G	E		I	N	T	R	O	D	.	
*	*	*													

B.- Introduction d'une donnée à virgule flottante à huit caractères (peut être négative) à partir de la troisième position de la première ligne de l'écran.
La légende "m/s." doit s'afficher à la fin de la donnée introduite.
La donnée sera ensuite stockée dans le registre à virgule flottante 500.

```
CLEAR           ;Efface la mémoire-tampon intermédiaire et
                ;place le pointeur sur la première position.
LOC     12      ;Positionne le pointeur d'affichage.
DISCH   109     ;Copie le caractère "m" dans la mémoire-
                ;tampon intermédiaire.
DISCH   47      ;Copie le caractère "/" dans la mémoire-
                ;tampon intermédiaire.
DISCH   115     ;Copie le caractère "s" dans la mémoire-
                ;tampon intermédiaire.
DISCH   46      ;Copie le caractère "." dans la mémoire-
                ;tampon intermédiaire.
LOC     2       ;Positionne le pointeur d'affichage.
COM     0       ;Affiche le contenu de la mémoire-tampon
                ;intermédiaire sur le LCD (m/s.).
INF     10      ;Permet d'introduire 10 caractères (l'un
                ;d'entre eux peut être le signe et l'autre la
                ;séparation décimale). Une fois introduits
                ;et après activation de la touche
                ;<ENTER>, le résultat est transféré dans la
                ;pile arithmétique.
STOF    500     ;Stocke le contenu de la pile arithmétique
                ;dans le registre à virgule flottante 500.
```

L'écran se présente de la manière suivante :

		*	*	*	*	*	*	*	*	*	*	m	/	s	.

C.- Introduction d'une donnée à virgule flottante à cinq caractères avec signe tout en utilisant le relais interne "Relais entrée" pour activer une sortie.

```
;Définition des libellés
texte0      lite    “APPUYEZ SUR F1”
texte1      lite    “INTROD. CODE"

        SETRI       600     500     ;Stocke la donnée 500 dans le registre
                                    ;entier 600.
        SETRI       300     6       ;Stocke la donnée 6 dans le registre
                                    ;entier 300.
        CLEAR                       ;Efface la mémoire-tampon intermédiaire et
                                    ;place le pointeur sur la première position.
        DISL        texte0          ;Copie, dans la mémoire-tampon
                                    ;intermédiaire, le message de la table de
                                    ;libellés "texte0".
        COM         0               ;Affiche le contenu de la mémoire-tampon
                                    ;intermédiaire sur le LCD.
        INK         356             ;Détecte l'activation d'une touche.
        RESET       104             ;Désactive le relais 104.
        JZ          fin             ;Saute à "fin" si la touche n'a pas été
                                    ;activée.
conti   CLEAR                       ;Efface la mémoire-tampon intermédiaire et
                                    ;place le pointeur sur la première position.
        DISL        texte1          ;Copie, dans la mémoire-tampon
                                    ;intermédiaire, le message de la table de
                                    ;libellés "texte1".
        COM         0               ;Affiche le contenu de la mémoire-tampon
                                    ;intermédiaire sur le LCD.
        INPFX       600     300     ;Permet d'introduire une donnée dont le
                                    ;format se trouve dans le registre entier
                                    ;300 et stocke la donnée introduite dans le
                                    ;registre entier désigné par le registre
                                    ;entier 600.
        LD          391             ;Charge l'état du relais 391 dans la pile
                                    ;logique (correspond au Relais entrée).
        OUT         104             ;Charge la valeur logique du relais 391 sur
                                    ;la sortie 104.
        INK         340             ;Détection d'une touche.
```

```
        JZ      conti       ;Saute à "conti" si la touche n'a pas été
                            ;activée.
fin     END
```

D.- Introduction d'une donnée à quatre caractères et à deux décimales, et stockage de la donnée dans le registre 500. Simultanément à l'introduction de la donnée, une sortie (105) envoie un train d'impulsions. Cette sortie se connecte grâce à l'entrée 1 et s'interrompt grâce à l'entrée 2. Observez également que le "Relais entrée" passe à 1 lorsque l'instruction INPCX est appelée et que la sortie continue à agir pendant l'exécution de cette instruction : le programme ne s'interrompt pas.

```
;Définition des libellés
texte1      lite    "INTROD. CODE"

        LD      1           ;Charge l'état de l'entrée 1 dans la pile
                            ;logique.
        LD      400         ;Charge l'état du relais 400 dans la pile
                            ;logique.
        ORLD                ;Effectue un OR logique entre l'entrée 1 et
                            ;le relais 400.
        ANDNT   2           ;Effectue un AND complémenté de l'entrée
                            ;2 avec le bloc logique précédent.
        OUT     400         ;Décharge le résultat de la fonction logique
                            ;effectuée sur le relais 400.
        LD      400         ;Charge l'état de l'entrée 400 dans la pile
                            ;logique.
        ANDNT   401         ;Effectue un AND complémenté de l'entrée
                            ;401 avec l'entrée précédente 400.
        TIM     250     10  ;Temporisation : 10 dixièmes de seconde.
        OUT     105         ;Décharge le résultat de la fonction logique
                            ;effectuée sur la sortie 105.
        LD      105         ;Charge l'état de l'entrée 105 dans la pile
                            ;logique.
        TIM     251     10  ;Temporisation : 10 dixièmes de seconde.
        OUT     401         ;Décharge le résultat de la fonction logique
                            ;effectuée sur la sortie du relais 401.
```

```
MOVCF   500         ;Charge la constante 500 dans la pile
                    ;arithmétique.
STOF    600         ;Stocke la constante 500 dans le registre
                    ;entier 600.
MOVCI   42          ;Charge la constante 42 dans la pile
                    ;arithmétique.
STOI    300         ;Stocke la constante 42 dans le registre
                    ;entier 300.
CLEAR               ;Efface la mémoire-tampon intermédiaire et
                    ;place le pointeur sur la première position.
DISL    texte1      ;Copie, dans la mémoire-tampon
                    ;intermédiaire, le message de la table de
                    ;libellés "texte1".
COM     0           ;Affiche le contenu de la mémoire-tampon
                    ;intermédiaire sur le LCD.
INPCX   600  300    ;Permet d'introduire une donnée dont le
                    ;format se trouve dans le registre entier
                    ;300 et stocke la donnée introduite dans le
                    ;registre entier désigné par le registre 300.
LD      391         ;Charge l'état du relais 391 dans la pile
                    ;logique (correspond au Relais entrée).
OUT     104         ;Sort la valeur par la sortie 104.
END
```

Les ***instructions*** d'affichage et de transmission via les ports de communication sont les suivantes :

CLEAR	**DISRI**	**DISIX**	**DISL**
LOC	**DISRF**	**DISFX**	**DISLX**
LOCX	**DISCH**	**DISCX**	**COM**

✓ MODE DE PROGRAMMATION

Les données pouvant être affichées ou transmises grâce au jeu d'instructions ci-dessus sont :

- ***Le contenu de registres entiers.***
- ***Le contenu de registres à virgule flottante.***
- ***Les caractères de la table ASCII.***
- ***Les messages de la table de libellés du programme.***

La mémoire-tampon intermédiaire disponible est de 132 octets. Toutes les instructions décrites fonctionnent sur cette mémoire-tampon. Seule l'instruction **COM** agit directement sur les écrans et les ports de communication.

L'opérande de l'instruction **COM** indique où le contenu de la mémoire-tampon intermédiaire doit être envoyé.

La programmation consiste ici à définir l'affichage dans la mémoire-tampon intermédiaire, puis à copier le résultat obtenu à l'endroit voulu.

✓ AFFICHAGE ET TRANSMISSION DE DONNÉES

Toutes les instructions d'affichage et de transmission fonctionnent sur la mémoire-tampon intermédiaire. Seule l'instruction **COM** agit directement sur les écrans et les ports de communication.

Le résultat des opérations effectuées dans la mémoire-tampon intermédiaire au moyen de ces instructions doit être copié sur les écrans ou les ports de communication grâce à l'instruction COM.

Le mode de programmation est identique pour les ports de communication et pour l'affichage.

Dans tous les équipements, chaque port de communication dispose d'un registre entier dans lequel figure la largeur de la transmission (consultez le Manuel Utilisateur de l'équipement correspondant).
La valeur figurant dans ce registre correspond au nombre de caractères qui seront transmis par le port de communication lors de l'exécution de l'instruction **COM**.
Si la valeur figurant dans le registre entier correspond à une largeur de transmission de 40, l'équipement charge cette donnée lors de chaque initialisation. Pour modifier cette valeur, il convient d'initialiser ce registre au début du programme.

L'équipement dispose de relais internes permettant de détecter l'état et l'expiration du délai d'attente (time-out) des ports de communication (consultez le Manuel Utilisateur de l'équipement correspondant).
Les relais varient selon les modèles d'équipements. Il peut s'agir de relais de :

- Réception : S'active en cas de réception.
- Transmission: S'active en cas de transmission.
- Signal CTS : Reste activé tant que le signal hardware CTS est activé.
  Indique que l'équipement est en état de transmettre.

- Expiration du délai d'attente (time-Out) : S'active en cas d'expiration du délai d'attente, c'est-à-dire 5 secondes après un essai de transmission infructueux.

Ces relais permettent d'exécuter des programmes capables, par exemple, de détecter si une imprimante est déconnectée et d'aviser l'utilisateur de l'existence d'une erreur en affichant un message lui proposant de faire un nouvel essai ou de quitter l'impression.
Pour de plus amples informations concernant l'adresse des écrans, consultez le Manuel Utilisateur de l'équipement correspondant.

L'équipement utilisera uniquement le nombre de caractères transmis accepté par l'écran. Les autres caractères seront négligés.
Exemple : supposons que le texte transmis à un écran à 8 caractères en comporte 16. Seuls les 8 premiers caractères des messages s'afficheront.
Certains équipements offrent la possibilité d'afficher des textes plus longs que la capacité de leur écran ne le leur permet normalement en faisant défiler horizontalement le texte.

COM

COM XXXX

MNÉMONIQUE : **COM**
OPÉRANDE XXXX : **Constante indiquant la destination de la copie provenant de la mémoire-tampon intermédiaire.**
CODE INSTRUCTION : **80**

DESCRIPTION :

COPIE de la mémoire-tampon intermédiaire sur les écrans et/ou les ports de communication.
L'opérande est la destination de la copie (consultez le Manuel Utilisateur de l'équipement correspondant).
Ne fait pas intervenir les piles (0).

☑ Exemple pour un équipement MIDA 64 :

```
CLEAR          ;Efface la mémoire-tampon intermédiaire et
               ;place le pointeur d'affichage sur la première
               ;position (la position 0).
DISL    1      ;Copie, dans la mémoire-tampon
               ;intermédiaire, le message occupant la
               ;première position de la table des libellés du
               ;programme.
COM     0      ;Copie le contenu de la mémoire-tampon
               ;intermédiaire sur le LCD.
COM     1      ;Copie également le contenu de la mémoire-
               ;tampon intermédiaire sur le port de
               ;communication COM1.
COM     2      ;Copie également le contenu de la mémoire-
               ;tampon intermédiaire sur le port de
               ;communication COM2.
COM     3      ;Et copie le contenu de la mémoire-tampon
               ;intermédiaire sur l'écran rouge à 6 caractères.
```

✓ POINTEUR DE LA MÉMOIRE-TAMPON INTERMÉDIAIRE

La gestion de l'emplacement des données dans la mémoire-tampon intermédiaire s'effectue grâce aux instructions CLEAR, LOC et LOCX.

Le pointeur (curseur) d'affichage est un registre interne réservé qui contient l'endroit où le prochain caractère introduit sera placé dans la mémoire-tampon intermédiaire.

Le pointeur d'affichage se déplace au fur et à mesure de l'exécution des instructions d'affichage et en fonction de la quantité de données enregistrées dans la mémoire-tampon intermédiaire.

REMARQUE : Tenez compte du fait que la première position de la mémoire-tampon intermédiaire est la position 0.

Les instructions ci-dessus permettent de modifier le pointeur pour contrôler l'affichage.

L'instruction **CLEAR** place le pointeur sur la première position de la mémoire-tampon intermédiaire après avoir effacé le contenu de celle-ci.

Les instructions **LOC** et **LOCX** placent simplement le curseur (pointeur d'affichage) à l'emplacement désigné par leurs opérandes.

L'opérande de l'instruction **LOC** indique la valeur absolue de l'endroit où le curseur sera placé.

L'opérande de l'instruction **LOCX** indique le registre dans lequel figure l'endroit où le curseur sera placé.

Vous trouverez, ci-après, une description des instructions de ce groupe :

CLEAR

MNÉMONIQUE : **CLEAR**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **70**

DESCRIPTION :

EFFACE le contenu de la mémoire-tampon intermédiaire et place le pointeur d'affichage sur la première position (position 0).
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
CLEAR       ;Efface la mémoire-tampon intermédiaire et
            ;place le pointeur d'affichage sur la première
            ;position située à gauche (la position 0).
COM     0   ;Efface le LCD.
COM     1   ;Envoie une ligne en blanc via COM1 (RS232).
COM     2   ;Envoie une ligne en blanc via COM2 (RS485).
```

							max. 132 caractères

(Mémoire-tampon intermédiaire)

LOC

LOC XXXX

MNÉMONIQUE : **LOC**
OPÉRANDE XXXX : **Constante indiquant la position du pointeur à l'intérieur de la mémoire-tampon intermédiaire.**
CODE INSTRUCTION : **75**

DESCRIPTION :

POSITIONNE le pointeur d'affichage en un point donné de la mémoire-tampon intermédiaire.
L'opérande est le point de la mémoire-tampon intermédiaire où le prochain caractère sera stocké.
La première position est la position 0.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
CLEAR          ;Efface la mémoire-tampon intermédiaire et
               ;place le pointeur d'affichage sur la première
               ;position (la position 0).
LOC     4      ;Place le pointeur d'affichage sur la quatrième
               ;position (la première est la position 0).
DISL    29     ;Copie, dans la mémoire-tampon intermédiaire, le
               ;message déclaré dans la position trente (la
               ;première est la position 0) de la table de libellés
               ;du programme.
COM     0      ;Copie le contenu de la mémoire-tampon
               ;intermédiaire sur le LCD.
```

				M	O	N		M	E	S	S	max. 132 caractères

⇧ (Mémoire-tampon intermédiaire)

LOCX

LOCX XXXX

MNÉMONIQUE : **LOCX**
OPÉRANDE XXXX : **Registre entier**
CODE INSTRUCTION : **76**

DESCRIPTION :

POSITIONNE le pointeur d'affichage en un point de la mémoire-tampon intermédiaire contenu dans un registre entier.
L'opérande est l'adresse du registre entier dans lequel figure le point de la mémoire-tampon intermédiaire où le prochain caractère sera stocké.
La première position est la position 0.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
MOVCI   1     ;Charge la constante 2 dans la pile arithmétique.
STOI    300   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 300.
CLEAR         ;Efface la mémoire-tampon intermédiaire et
              ;place le pointeur d'affichage sur la première
              ;position (la position 0).
LOCX    300   ;Place le pointeur d'affichage sur la première
              ;position (la première est la position 0).
DISL    29    ;Copie, dans la mémoire-tampon intermédiaire, le
              ;message déclaré dans la position trente (la
              ;première est la position 0) de la table de libellés
              ;du programme.
COM     0     ;Copie le contenu de la mémoire-tampon
              ;intermédiaire sur le LCD.
```

	A	U	T	R	E		M	E	S	S	A	max. 132 caractères

⇧ (Mémoire-tampon intermédiaire)

✓ MESSAGES ET CARACTÈRES ASCII

Les instructions **DISL** et **DISLX** permettent d'afficher et de transmettre les messages de ce type. Le MIDA dispose d'une table contenant un certain nombre de messages qui comportent eux-mêmes un certain nombre de caractères (consultez le Manuel Utilisateur).

Ces messages doivent être déclarés dans le programme lui-même, au moyen de la pseudo-instruction **LITE** si vous employez un éditeur standard pour programmer l'équipement.

Un certain nombre de caractères est assigné à chaque message, quelle que soit la longueur réelle de celui-ci. Cela signifie que l'affichage d'un message occupera la longueur maximale qui lui est assignée, que ce message comporte 1, 2 ou 3 caractères. Le reste sera complété par des espaces en blanc.

Les messages peuvent s'enchaîner les uns aux autres dans la mémoire-tampon intermédiaire pour former des textes plus longs.

Les instructions **LOC** et **LOCX** permettent de modifier l'emplacement du pointeur d'affichage pour intercaler des données numériques, des caractères complémentaires, etc. à l'intérieur d'un message.

L'opérande de l'instruction **DISL** indique la position du message, en valeur absolue, à l'intérieur de la table des libellés. La première position est la position 0.

L'opérande de l'instruction **DISLX** indique le registre entier dans lequel figure la position du message à l'intérieur de la table.

Vous pouvez également afficher individuellement les caractères ASCII au moyen de l'instruction **DISCH.**

L'instruction **DISCX** contient, quant à elle, un opérande indiquant le registre entier qui contient le caractère ASCII à enregistrer dans la mémoire-tampon intermédiaire.

Pour de plus amples informations concernant la table de caractères ASCII disponible pour chaque équipement et pour chaque type d'écran, consultez le Manuel Utilisateur correspondant.

Si la longueur du message envoyé à la mémoire-tampon intermédiaire est supérieure au nombre de caractères disponibles sur l'écran, celui-ci affiche les premiers caractères et ignore les suivants.

Vous trouverez, ci-après, une description des instructions de ce groupe :

DISL

DISL XXXX

MNÉMONIQUE : **DISL**
OPÉRANDE XXXX : **Constante indiquant l'adresse du message à l'intérieur de la table des libellés.**
CODE INSTRUCTION : **71**

DESCRIPTION :

COPIE un message de la table de libellés du programme dans la mémoire-tampon intermédiaire.
L'opérande est l'adresse du message à l'intérieur de la table des libellés.
Le premier message de la table est le message 0.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
Texte1          lite    "Message Un"
Texte2          lite    "Message Deux"
Texte3          lite    "Message Trois"
Texte4          lite    "Message Quatre"
Texte5          lite    "Message Cinq"

        CLEAR           ;Efface la mémoire-tampon intermédiaire et place
                        ;le pointeur sur la première position (0).
        DISL    3       ;Copie le texte numéro 4 (adresse 3) de la table
                        ;dans la mémoire-tampon intermédiaire et le place
                        ;sur la première position de celle-ci (l'adresse du
                        ;premier message est l'adresse 0).
        COM     0       ;Copie le message "Message Quatre" de la
                        ;mémoire-tampon intermédiaire sur le LCD.
```

DISLX

DISLX XXXX

MNÉMONIQUE : **DISLX**
OPÉRANDE XXXX : **Registre entier**
CODE INSTRUCTION : **72**

DESCRIPTION :

COPIE, dans la mémoire-tampon intermédiaire, le message signalé.
L'opérande est le registre entier qui contient l'adresse du message à l'intérieur de la table de libellés du programme.
Le premier message de la table est le message 0.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
Texte1          lite    "Message Un"
Texte2          lite    "Message Deux"
Texte3          lite    "Message Trois"
Texte4          lite    "Message Quatre"
Texte5          lite    "Message Cinq"

        MOVCI   2       ;Charge la constante 2 dans la pile arithmétique.
        STOI    410     ;Stocke le contenu de la pile arithmétique dans le
                        ;registre entier 410.
        CLEAR           ;Efface la mémoire-tampon intermédiaire et
                        ;place le pointeur d'affichage sur la première
                        ;position (la position 0).
        DISLX   410     ;Copie le texte numéro 3 (adresse 2) de la table
                        ;des libellés dans la mémoire-tampon
                        ;intermédiaire (l'adresse du premier message est
                        ;l'adresse 0).
        COM     0       ;Copie le message "Message Trois" de la
                        ;mémoire-tampon intermédiaire sur le LCD.
```

DISCH

DISCH XXXX

MNÉMONIQUE : **DISCH**
OPÉRANDE XXXX : **Valeur décimale du caractère ASCII.**
CODE INSTRUCTION : **77**

DESCRIPTION :

COPIE, dans la mémoire-tampon intermédiaire, le caractère ASCII signalé.
L'opérande est la valeur décimale du caractère ASCII correspondant.
Pour déterminer quels caractères ASCII sont compatibles avec chaque équipement, consultez le Manuel Utilisateur correspondant.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
CLEAR          ;Efface la mémoire-tampon intermédiaire
               ;et place le pointeur d'affichage sur la
               ;première position.
DISCH    79    ;Copie le caractère “O” dans la mémoire-tampon
               ;intermédiaire.
DISCH    75    ;Copie le caractère “K” dans la mémoire-tampon
               ;intermédiaire.
COM      1     ;Copie le contenu de la mémoire-tampon
               ;intermédiaire sur le port de communication.
```

DISCX

DISCX XXXX

MNÉMONIQUE : **DISCX**
OPÉRANDE XXXX : **Registre entier**
CODE INSTRUCTION : **81**

DESCRIPTION :

COPIE, dans la mémoire-tampon intermédiaire, le caractère ASCII indiqué par l'opérande.
L'opérande XXXX est le registre entier qui contient la valeur décimale du caractère ASCII à transférer.
Pour déterminer quels caractères ASCII sont compatibles avec chaque équipement, consultez le Manuel Utilisateur correspondant.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
MOVCI   79    ;Charge la constante 79 dans la pile
              ;arithmétique.
STOI    200   ;Stocke le contenu de la pile arithmétique dans le
              ;registre entier 200.
CLEAR         ;Efface la mémoire-tampon intermédiaire et
              ;place le pointeur d'affichage sur la première
              ;position (la position 0).
DISCX   200   ;Copie le caractère ASCII 79 (O) dans la
              ;mémoire-tampon intermédiaire, car c'est le
              ;caractère signalé par le registre entier 200.
COM     0     ;Copie le contenu de la mémoire-tampon
              ;intermédiaire sur l'écran.
```

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.-

```
;DÉFINITION DES LIBELLÉS
texte1      lite    "Ce texte contient"
texte2      lite    "32 caractères"
texte3      lite    "s"
;
        MOVCI       33      ;Charge la constante 33 dans la pile arithmétique.
        STOI        20      ;Stocke le contenu de la pile arithmétique dans le
                            ;registre entier 20 (registre correspondant à la
                            ;largeur de transmissions voulue pour COM1).
        MOVCI       16      ;Charge la constante 16 dans la pile arithmétique.
        STOI        21      ;Stocke le contenu de la pile arithmétique dans le
                            ;registre entier 21 (registre correspondant à la
                            ;largeur de transmissions voulue pour COM2).
        CLEAR               ;Efface la mémoire-tampon intermédiaire et place
                            ;le pointeur sur la première position.
        DISL        texte1  ;Charge le message de la table des libellés
                            ;signalé dans la mémoire-tampon intermédiaire.
        DISL        texte2  ;Charge le message de la table des libellés
                            ;signalé dans la mémoire-tampon intermédiaire.
        DISL        texte3  ;Charge le message de la table des libellés
                            ;signalé dans la mémoire-tampon intermédiaire.
        COM         1       ;Transmission de la mémoire-tampon temporaire
                            ;à COM1.
        COM         2       ;Transmission de la mémoire-tampon temporaire
                            ;à COM1.
```

Résultat :

Ce texte contient 32 caractères

(Transmission via COM1)

Ce texte contient

(Transmission via COM2)

B.-

```
;DÉFINITION DES LIBELLÉS
;
texte_A     lite    “Ceci est le texte1”
texte_B     lite    “Alarme tension”
;
;PROGRAMME
        ...
        ...
    MOVCI       1           ;Charge la constante 1 dans la pile
                            ;arithmétique.
    STOI        500         ;Stocke le contenu de la pile
                            ;arithmétique dans le registre entier 500.
    CLEAR                   ;Efface la mémoire-tampon intermédiaire et
                            ;place le pointeur sur la première position.
    DISL        texte_A     ;Charge le message de la table des libellés
                            ;signalé dans la mémoire-tampon
                            ;intermédiaire.
    DISLX       500         ;Copie le texte numéro 2 (adresse 1) de la
                            ;table des libellés dans la mémoire-tampon
                            ;intermédiaire (l'adresse du premier
                            ;message est l'adresse 0).
    COM         0           ;Affiche le contenu de la mémoire-tampon
                            ;intermédiaire sur le LCD.
```

Résultat :

C	e	c	i		e	s	t		l	e		t

e	x	t	1		A	l	a	r	m	e		t

e	n	s	i	o	n	max. 132 caractères

(Mémoire-tampon intermédiaire)

✓ AFFICHAGE ET TRANSMISSION DE DONNÉES NUMÉRIQUES

Il est possible d'afficher et de transmettre des données numériques entières ou à virgule flottante.

Les instructions **DISRI** et **DISRF** permettent d'effectuer ces opérations et disposent pour ce faire de deux opérandes.

Le deuxième opérande de ces instructions indique le format de la donnée, c'est-à-dire son nombre de caractères et de décimales, ainsi que le type de justification (gauche ou droite).

Il est important de tenir compte du format d'introduction des données entrées par l'intermédiaire du clavier au moment de les afficher.

Le signe moins (-) s'affiche, mais pas le signe plus (+). La valeur se déplace donc d'un espace vers la gauche et il y a un espace en blanc lorsque la donnée est positive.

Si le format d'affichage choisi pour les données entières est 0, l'équipement affiche la donnée en utilisant tous les caractères nécessaires à sa représentation.

Si le format d'affichage choisi pour les données à virgule flottante est 0, l'équipement affiche la donnée en utilisant tous les caractères nécessaires à sa représentation sans décimale.

L'équipement réserve l'espace nécessaire à l'affichage dans la mémoire-tampon intermédiaire en fonction du format programmé et laisse le reste des caractères en blanc. Si la donnée est plus longue que le format (erreur, etc.), l'équipement affiche toujours celle-ci en utilisant les caractères nécessaires pour ne pas fausser l'affichage.

Si vous utilisez des format à un seul caractère pour l'affichage de données à virgule flottante, aucune décimale ne s'affichera.

Évitez d'employer des formats d'affichage inutilisables ou incorrects, car cela risquerait de produire des effets indésirables au niveau de l'affichage général de l'équipement. Le compilateur ne détecte pas les formats non autorisés.

Les instructions **DISIX** et **DISFX** fonctionnent comme les précédentes mais, contrairement à celles-ci, elles sont indexées (voir compatibilité de format avec les instructions **INPIX** et **INPFX** dans le chapitre "Reconnaissance du clavier" de ce manuel).

Vous trouverez, ci-après, une description des instructions de ce groupe :

DISRI

DISRI XXXX YYYY

MNÉMONIQUE : **DISRI**
OPÉRANDE XXXX : **Registre entier**
OPÉRANDE YYYY : **Constante indiquant le format dans lequel sera effectuée la copie.**
CODE INSTRUCTION : **73**

DESCRIPTION :

COPIE le contenu d'un registre entier dans la mémoire-tampon intermédiaire.
L'opérande XXXX est l'adresse du registre entier à copier.
L'opérande YYYY est le format de la donnée à copier.
Selon l'équipement que vous vous disposez à programmer, le format peut être AB ou ABC :
où AB est le nombre de caractères à copier.
où C indique la position de la séparation décimale.
Si le deuxième opérande est positif, l'affichage est justifié à droite.
Si le deuxième opérande est négatif, l'affichage est justifié à gauche.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
CLEAR          ;Efface la mémoire-tampon intermédiaire et
               ;place le pointeur d'affichage sur la première
               ;position (la position 0).
DISRI 300  -3  ;Copie le contenu du registre entier 300 (total : 3
               ;caractères, signe compris) dans la mémoire-
               ;tampon intermédiaire. Le format étant négatif,
               ;l'affichage sera justifié à gauche.
COM  0         ;Copie le contenu de la mémoire-tampon
               ;intermédiaire sur le LCD.
```

DISIX

DISIX XXXX YYYY

MNÉMONIQUE : **DISIX**
OPÉRANDE XXXX : **Registre entier**
OPÉRANDE YYYY : **Registre entier**
CODE INSTRUCTION : **74**

DESCRIPTION :

COPIE, dans la mémoire-tampon intermédiaire, le contenu du registre entier signalé.
L'opérande XXXX est le registre entier qui contient l'adresse du registre entier à copier.
L'opérande YYYY est l'adresse du registre entier qui contient le format de la donnée à copier.
Selon l'équipement que vous vous disposez à programmer, le format peut être AB ou ABC :
où AB est le nombre de caractères à copier,
où C indique la position de la séparation décimale.
Si le format est positif, l'affichage est justifié à droite.
Si le format est négatif, l'affichage est justifié à gauche.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
SETRI   300   500   ;Stocke la donnée 500 dans le reg. entier 300.
SETRI   301   3     ;Stocke la donnée 3 dans le reg. entier 301.
CLEAR               ;Efface la mémoire-tampon intermédiaire et
                    ;place le pointeur d'affichage sur la position 0
DISIX   300   301   ;Copie, dans la mémoire-tampon
                    ;intermédiaire, le contenu du registre entier
                    ;500 (registre désigné par le registre entier
                    ;300) dans un format à 3 caractères, signe
                    ;non compris, et justifié à droite (format
                    ;indiqué par registre entier 301).
COM     0           ;Copie le contenu de la mémoire-tampon
                    ;intermédiaire sur le LCD.
```

DISRF

DISRF XXXX YYYY

MNÉMONIQUE : **DISRF**
OPÉRANDE XXXX : **Registre à virgule flottante**
OPÉRANDE YYYY : **Constante indiquant le format dans lequel sera effectuée la copie.**
CODE INSTRUCTION : **68**

DESCRIPTION :

COPIE le contenu d'un registre à virgule flottante dans la mémoire-tampon intermédiaire.
L'opérande XXXX est l'adresse du registre à virgule flottante à copier.
L'opérande YYYY est le format à copier, en nombre de caractères et de décimales. Le format est : ABC
où AB est le nombre de caractères à afficher, signe et séparation décimale compris, et où C est le nombre de décimales.
Si le format est positif, l'affichage est justifié à droite.
Si le format est négatif, l'affichage est justifié à gauche.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
MOVCF   -28.5       ;Charge la constante à virgule flottante
                    ;-28.5 dans la pile arithmétique.
STOF    100         ;Stocke le contenu de la pile arithmétique
                    ;dans le registre à virgule flottante 100.
CLEAR               ;Efface la mémoire-tampon intermédiaire et
                    ;place le pointeur d'affichage sur la
                    ;première position (la position 0).
DISRF   100  73     ;Copie, dans la mémoire-tampon
                    ;intermédiaire, le contenu du registre à
                    ;virgule flottante 100 qui comportera 7
                    ;caractères, dont 3 décimales. Justification
                    ;à droite.
COM     0           ;Copie le contenu de la mémoire-tampon
                    ;intermédiaire sur le LCD.
```

DISFX

DISFX XXXX YYYY

MNÉMONIQUE : **DISFX**
OPÉRANDE XXXX : **Registre entier**
OPÉRANDE YYYY : **Registre entier**
CODE INSTRUCTION : **69**

DESCRIPTION :

COPIE, dans la mémoire-tampon intermédiaire, le contenu du registre à virgule flottante désigné.
L'opérande XXXX est le registre entier qui contient l'adresse du registre à virgule flottante à copier.
L'opérande YYYY est l'adresse du registre entier qui contient le format à copier. Le format est : ABC
où AB est le nombre de caractères à afficher, signe et séparation décimale compris, et où C est le nombre de décimales.
Si le format est positif, l'affichage est justifié à droite.
Si le format est négatif, l'affichage est justifié à gauche.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
SETRI   500  100   ;Stocke la donnée 100 dans le registre
                   ;entier 500.
SETRI   301  62    ;Stocke la donnée 62 dans le registre
                   ;entier 301.
MOVCF   33.1578    ;Charge la constante à virgule flottante
                   ;33.1578 dans la pile arithmétique.
STOF    100        ;Stocke le contenu de la pile
                   ;arithmétique dans le registre à virgule
                   ;flottante 100.
CLEAR              ;Efface la mémoire-tampon intermédiaire et
                   ;place le pointeur d'affichage sur la
                   ;première position (la position 0).
```

```
DISFX     500   301   ;Copie, dans la mémoire-tampon
                      ;intermédiaire, le contenu du registre à
                      ;virgule flottante 100 (registre désigné par
                      ;le registre entier 500) dans un format à 6
                      ;caractères (signe non compris) dont 2
                      ;décimales, et justifié à droite (format
                      ;indiqué par registre entier 301).
COM       0           ;Copie le contenu de la mémoire-tampon
                      ;intermédiaire sur le LCD.
```

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Affichage d'un registre entier à l'aide de deux méthodes, en utilisant les instructions DISRI et DISIX sur un même écran :

```
MOVCI   -1435       ;Charge la constante -1435 dans la
                    ;pile arithmétique.
STOI    500         ;Stocke le contenu de la pile
                    ;arithmétique dans le registre entier 500.
MOVCI   500         ;Charge la constante 500 dans la pile
                    ;arithmétique.
STOI    501         ;Stocke le contenu de la pile
                    ;arithmétique dans le registre entier 501.
MOVCI   5           ;Charge la constante 5 dans la pile
                    ;arithmétique.
STOI    502         ;Stocke le contenu de la pile
                    ;arithmétique dans le registre entier 502.
CLEAR               ;Efface la mémoire-tampon intermédiaire et
                    ;place le pointeur sur la première position.
DISRI   500   5     ;Copie, dans la mémoire-tampon
                    ;intermédiaire, le contenu du registre entier
                    ;500 dans un format à 5 caractères.
LOC     16          ;Place le pointeur sur la position 16.
DISIX   501   502   ;Copie le contenu du registre 501 dans la
                    ;mémoire-tampon intermédiaire, dans le
                    ;format qui figure dans le registre 502.
COM     0           ;Affiche le contenu de la mémoire-tampon
                    ;intermédiaire sur le LCD.
```

Résultat :

-	1	4	3	5											
-	1	4	3	5											

(Afficheur LCD)

B.- Affichage d'un registre à virgule flottante à l'aide de deux méthodes, en utilisant les instructions DISRF et DISFX sur un même écran :

```
MOVCF   -1435.5      ;Charge la constante –1435.5 dans la pile
                     ;arithmétique.
STOF    500          ;Stocke le contenu de la pile arithmétique
                     ;dans le registre à virgule flottante 500.
MOVCF   500          ;Charge la constante 500 dans la pile
                     ;arithmétique.
STOF    501          ;Stocke le contenu de la pile arithmétique
                     ;dans le registre à virgule flottante 501.
MOVCF   71           ;Charge la constante 71 dans la pile
                     ;arithmétique.
STOF    502          ;Stocke le contenu de la pile arithmétique
                     ;dans le registre à virgule flottante 502.
CLEAR                ;Efface la mémoire-tampon intermédiaire et
                     ;place le pointeur sur la première position.
DISRF   500  71      ;Copie le contenu du registre 500 dans la
                     ;mémoire-tampon intermédiaire, dans le
                     ;format signalé par le deuxième opérande
                     ;71.
LOC     16           ;Place le pointeur dans la position
                     ;indiquée.
DISFX   501  502     ;Copie le contenu du registre 501 dans la
                     ;mémoire-tampon intermédiaire, dans le
                     ;format signalé par le contenu du registre
                     ;502.
COM     0            ;Affiche le contenu de la mémoire-tampon
                     ;intermédiaire sur le LCD.
```

Résultat :

-	1	4	3	5	.	5									
-	1	4	3	5	.	5									

(Afficheur LCD)

C.- Affichage :
- d'un texte,
- du contenu d'un registre à virgule flottante à deux décimales et dont la valeur peut varier entre 0 et 327.67, avec justification à gauche, et
- d'un caractère ASCII sur le LCD.
- et transmission d'un message de 32 caractères via les deux ports de communication.

```
;DÉFINITION DES ÉTIQUETTES
;
texte1      lite        "MOITIÉ"
texte2      lite        "- - - - - - - - - - - - - - - "
texte3      lite        "V="
;
        CLEAR                       ;Efface la mémoire-tampon intermédiaire et
                                    ;place le pointeur sur la première position.
        DISL        texte3          ;Copie, dans la mémoire-tampon
                                    ;intermédiaire, le texte de la table de libellés
                                    ;signalé.
        LOC         2               ;Place le pointeur dans la position indiquée.
        DISRF       500   -62       ;Copie le contenu du reg. à virgule flottante
                                    ;500 dans la mémoire-tampon
                                    ;intermédiaire dans le format signalé par le
                                    ;deuxième opérande.
        LOC         5               ;Place le pointeur dans la position indiquée.
        DISCH       65              ;Copie le caractère ASCII "A" dans la
                                    ;mémoire-tampon intermédiaire.
        COM         0               ;Affiche le contenu de la mémoire-tampon
                                    ;intermédiaire sur le LCD.
        CLEAR                       ;Efface la mémoire-tampon intermédiaire et
                                    ;place le pointeur sur la première position.
        DISL        texte1          ;Copie, dans la mémoire-tampon
                                    ;intermédiaire, le texte de la table de
                                    ;libellés signalé.
        DISL        texte2          ;Copie, dans la mémoire-tampon
                                    ;intermédiaire, le texte de la table de
                                    ;libellés signalé.
        COM         1               ;Transmet le contenu de la mémoire-
                                    ;tampon intermédiaire via le port de comm.
```

```
        COM     2       ;Transmet le contenu de la mémoire-
                        ;tampon intermédiaire via le port de comm.
```

D.- Affichage :

- d'un texte signalé par un registre et
- d'une variable se trouvant à l'intérieur de ce texte et dont la décimale est comprise entre 0 et 99.9, avec justification à gauche,
- Transmission de la date et de l'heure via le port de comm. RS232.

```
;DÉFINITION DES ÉTIQUETTES
;
texte32     lite        "VALEUR:"
;
        MOVCI   0       ;Charge la constante 0 dans la pile
                        ;arithmétique.
        STOI    500     ;Stocke le contenu de la pile arithmétique
                        ;dans le registre entier 500.
        CLEAR           ;Efface la mémoire-tampon intermédiaire et
                        ;place le pointeur sur la première position.
        DISLX   500     ;Copie le texte indiqué par le contenu du
                        ;registre entier 500 dans la mémoire-
                        ;tampon intermédiaire.
        LOC     9       ;Place le pointeur dans la position indiquée
        DISRF   501 -41 ;Copie, dans la mémoire-tampon
                        ;intermédiaire, le contenu du registre entier
                        ;501 dans le format signalé par le
                        ;deuxième opérande.
        COM     0       ;Affiche le contenu de la mémoire-tampon
                        ;intermédiaire sur le LCD.
        CLEAR           ;Efface la mémoire-tampon intermédiaire et
                        ;place le pointeur sur la première position.
        TIME            ;Copie l'heure au format HH:MM:SS dans
                        ;la mémoire-tampon intermédiaire.
        DISCH   32      ;Copie le caractère ASCII ”espace en
                        ;blanc” dans la mémoire-tampon
                        ;intermédiaire.
        DATE            ;Copie la date au format JJ:MM:AA dans
                        ;la mémoire-tampon intermédiaire.
        COM     1       ;Transmet le contenu de la mémoire-
                        ;tampon intermédiaire via le port COM1.
```

✓ AFFICHAGE EN MODE DÉFILEMENT

Certains modèles d'équipements MIDA offrent la possibilité d'afficher les textes, les messages et/ou toute représentation des relais en les faisant défiler horizontalement.

Les équipements de ce type possèdent un relais "Affichage en mode Défilement" (consultez la topographie de la mémoire dans le Manuel Utilisateur de l'équipement).

Lorsque l'instruction COM x (x est l'écran sur lequel les données sont adressées) est exécutée, l'affichage du contenu de la mémoire-tampon intermédiaire est fixe si l'état du relais "Affichage en mode Défilement" est "0" et défile de droite à gauche sur l'écran si l'état de ce relais est "1".

Pour définir la longueur du texte à afficher, il vous suffit de placer un caractère ASCII "0" à la fin du texte. La longueur maximale du texte est la longueur maximale admise par l'équipement. Tout caractère supplémentaire est négligé.

Les instructions de reconnaissance de clavier et les commandes du MIDAbus qui affectent les écrans ne tiennent pas compte de l'état du relais "Affichage en mode Défilement".

La modification de l'état du relais ne change pas l'affichage. La seule manière de modifier le comportement de l'écran (et, si nécessaire, son contenu) est d'exécuter l'instruction COM.

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.-

```
;Définition des étiquettes
defil        equ   xx (indiquer le relais "Affichage en mode
                       Défilement").
;
;Définition des libellés
texte1       lite    “UN TEXTE DE 24 CARACT."
```

```
.....
        CLEAR                   ;Efface la mémoire-tampon intermédiaire
                                ;et place le pointeur d'affichage sur la
                                ;première position.
        DISL        texte1      ;Copie le "texte1" de la table des libellés
                                ;dans la mémoire-tampon intermédiaire
                                ;et le place dans la première position de
                                ;celle-ci.
        LOC         22          ;Place le pointeur d'affichage sur la
                                ;position 23 (la première est la position 0).
        DISCH       0           ;Copie le caractère “0” dans la mémoire-
                                ;tampon intermédiaire.
        SET         defil.      ;Active le relais "Affichage en mode
                                "Défilement".
        COM         0           ;Copie le contenu de la mémoire-tampon
                                ;intermédiaire sur le LCD.
```

Sur un écran à 8 caractères, le texte “UN TEXTE DE 24 CARACT.” se présentera de la manière suivante :

							U
						U	N
					U	N	
				U	N		T

C	A	R	A	C	T	.	U
A	R	A	C	T	.	U	N
R	A	C	T	.	U	N	
A	C	T	.	U	N		T

Étant donné que le message passe en boucle, la fin du texte s'enchaîne avec le début ("CARACT.UN TEXTE DE 24"). Pour une meilleure lisibilité, il est donc conseillé d'ajouter un caractère de séparation entre le début et la fin du texte (par exemple un espace en blanc).

B.-

Supposons que nous soyons en train de contrôler un processus, que le registre entier 250 corresponde au temps écoulé (en minutes) et le registre entier 260 à la température (en dixièmes de degrés). Nous souhaitons afficher un texte défilant indiquant le temps écoulé et la température, et si la température est supérieure à 50º, afficher le message "ALARME".

```
;Définition des étiquettes
temps       equ   xxx    ;(registre entier)
temp        equ   xxx    ;(registre entier)
degres      equ   xxx    ;(registre entier)
defil       equ   xxx    ;(relais "Affichage en mode Défilement")
;
;Définition des libellés
txt         lite  "TEMPS XXX MIN. TEMPÉRATURE XX.X "
txt_ala     lite  "ALARME"

        MOVRI   temp
        MOVCI   500             ;50º (en dixièmes de degrés).
        CLEAR
        CPGI    ala             ;Saute en cas d'alarme (TEMP>50º)
        DISL    txt             ;Copie le texte dans la mémoire-
                                ;tampon.
        LOC     6
        DISRI   temps   30
        LOC     27
        DISRI   degres  41
        LOC     34
        DISCH   0               ;Copie le caractère 0 à la fin.
        SET     defil           ;Active le relais "Affichage en
                                ;mode Défilement".
        COM     0
        .....
ala     RESET   defil           ;Désactive le relais "Affichage en
                                ;mode Défilement".
        CLEAR
        DISL    txt_ala
        COM     0
        .....
```

Les équipements MIDA disposent d'une horloge en temps réel protégée par une batterie au Ni-Cd.

Ils possèdent des registres entiers de 16 bits dont le contenu, qui est le reflet de la date/l'heure de l'horloge interne, est automatiquement actualisé.
Des opérations arithmétiques peuvent être appliquées à ces registres comme à tout autre registre entier de l'équipement.
Un programme MIDA peut donc accéder aux registres suivants :

DONNÉE	REGISTRE
Secondes écoulées	Indique le nombre de secondes écoulées à l'intérieur de la minute en cours. **(0 - 59)**
Minutes écoulées	Indique le nombre de minutes écoulées à l'intérieur de l'heure en cours. **(0 - 59)**
Heures écoulées	Indique l'heure du jour en cours. **(0 - 23)**
Horaire	Indique le nombre de minutes écoulées depuis 0 heure (nombre de minutes du jour). **(heure * 60 + minute)**
Jour du mois	Indique le jour du mois. **(1 - 31)**
Jour de la semaine	Indique le jour de la semaine. **(0 - 6, lundi – dimanche)**
Mois de l'année	Indique le mois de l'année. **(1 - 12)**
Année	Indique l'année. **(les deux derniers chiffres)**

✓ MODE DE PROGRAMMATION

Le programme peut accéder à tous les registres décrits et leur appliquer des opérations arithmétiques, les afficher, les comparer, les stocker dans les registres de la base de données, etc.
Les principales fonctions pour lesquelles l'horloge en temps réel est employée sont les suivantes :

- Exécution programmée dans le temps.
- Affichage et/ou transmission via port série.
- Mise à l'heure de l'horloge.

Exécution programmée dans le temps : ce type de fonction permet d'exécuter un processus à une heure, un jour de la semaine, etc. donnés.
Il convient d'employer les registres mentionnés précédemment pour effectuer les comparaisons et/ou les opérations arithmétiques nécessaires (reportez-vous aux chapitres Instructions arithmétiques et Comparaisons arithmétiques de ce manuel).

Affichage et/ou transmission via port série : vous pouvez utiliser les instructions d'affichage et de communication de registre décrites dans le paragraphe Affichage et communications ou les instructions spéciales dont dispose éventuellement votre équipement.
Il existe des instructions spécialement conçues pour l'affichage ou la transmission de données via les ports série (imprimante, PC, etc.). Il s'agit des instructions suivantes :

DATE	**TIME**

Mise à l'heure de l'horloge de l'équipement : les instructions permettant d'effectuer cette opération sont :

CLOCK	**CLKP**

DATE

MNÉMONIQUE : **DATE**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **78**

DESCRIPTION :

COPIE la date au format JJ/MM/AA dans la mémoire-tampon intermédiaire pour afficher la date de l'horloge interne de l'équipement sur le(s) écran(s) ou la transmettre via n'importe quel port de communication.
Cette instruction peut également être utilisée pour stocker la date dans l'un des fichiers de la base de données (si l'équipement en possède une).
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
CLEAR       ;Efface la mémoire-tampon intermédiaire et
            ;place curseur sur la première position (la position
            ;0).
DATE        ;Copie la date au format JJ/MM/AA dans la
            ;mémoire-tampon intermédiaire.
COM     0   ;Copie le contenu de la mémoire-tampon
            ;intermédiaire sur le LCD.
```

TIME

MNÉMONIQUE : **TIME**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **79**

DESCRIPTION :

COPIE l'heure au format HH:MM:SS dans la mémoire-tampon intermédiaire pour afficher l'heure de l'horloge interne de l'équipement sur le(s) écran(s) ou la transmettre à n'importe quel port de communication.
Cette instruction peut également être utilisée pour stocker l'heure dans l'un des fichiers de la base de données (si l'équipement en possède une).
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
CLEAR          ;Efface la mémoire-tampon intermédiaire et
               ;place le pointeur d'affichage sur la première
               ;position (la position 0).
TIME           ;Copie l'heure au format HH:MM:SS dans la
               ;mémoire-tampon intermédiaire.
COM      0     ;Copie le contenu de la mémoire-tampon
               ;intermédiaire sur le LCD.
```

CLOCK

MNÉMONIQUE : **CLOCK**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **84**

DESCRIPTION :

Permet d'ACTUALISER l'HEURE/la DATE de l'horloge interne de l'équipement par l'intermédiaire du clavier.
L'exécution de cette instruction entraîne l'interruption de l'exécution du programme et l'affichage de la date, de l'heure et du jour de la semaine sur le LCD.
Ces données étant affichées, vous pouvez actualiser l'heure/la date.
Pour modifier l'heure/la date, appuyez sur la touche <CLEAR>.
L'exécution de cette instruction s'achève par l'activation de la touche <ENTER>.
Ne fait pas intervenir les piles (0).
Interrompt l'exécution du programme.

☑ Exemple :

```
        INK   340   ;Détection d'une touche.
        JZ    eti1  ;Si la touche a été activée, le programme passe à
                    ;la ligne suivante. Dans le cas contraire, il saute à
                    ;"éti1".
        CLOCK       ;Exécution de la fonction d'ACTUALISATION de
                    ;l'HEURE/la DATE et interruption de l'exécution du
                    ;programme qui ne sera relancée que lorsque
                    ;vous appuierez sur la touche
                    ;<ENTER>(ENTRÉE).
        ......
eti1    ......
```

CLKP

CLKP

MNÉMONIQUE : **CLKP**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **97**

DESCRIPTION :

Permet d'ACTUALISER l'HEURE/la DATE de l'horloge interne de l'équipement par l'intermédiaire du clavier, mais sans interrompre le programme utilisateur.
L'instruction CLKP ne sera pas exécutée si l'une des instructions INPIX, INPFX ou INPCX est en cours d'exécution.
L'exécution de cette instruction n'entraîne pas l'interruption de l'exécution du programme. La date, l'heure et le jour de la semaine s'affichent sur le LCD.
Ces données étant affichées, vous pouvez actualiser l'heure/la date.
Pour modifier l'heure/la date, appuyez sur la touche <CLEAR>
L'exécution de cette instruction s'achève par l'activation de la touche <ENTER>.
Un relais interne, le "Relais entrée" (consultez le Manuel Utilisateur de l'équipement) s'active lorsque vous appelez cette instruction. Il reste activé pendant l'introduction de la donnée et se désactive lorsque vous validez celle-ci au moyen de la touche <ENTER>.
Le contenu de l'écran peut être modifié pendant l'introduction de l'heure/la date. Il sera actualisé lorsque la nouvelle heure/date aura été validée.
Ne fait pas intervenir les piles (0).
N'interrompt pas l'exécution du programme.

☑ Exemple :

```
        INK   30    ;Détection d'une touche.
        JZ    eti1  ;Si la touche a été activée, le programme passe à la ligne
                    ;suivante. Dans le cas contraire, il saute à "éti1".
        CLKP        ;Exécution de la fonction d'ACTUALISATION de l'horloge
                    ;et activation du "Relais entrée". Poursuite le programme.
eti1    ......
```

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Affichage de l'heure et de la date au début de chaque ligne du LCD, et d'un message OK à la fin de la première ligne ; transmission de la date et de l'heure séparées par un espace en blanc via COM1.

```
CLEAR          ;Efface la mémoire-tampon intermédiaire et
               ;place le pointeur sur la première position.
TIME           ;Copie l'heure dans la mémoire-tampon intermédiaire.
LOC     16     ;Place le pointeur dans la position indiquée.
DATE           ;Copie la date dans la mémoire-tampon intermédiaire.
LOC     14     ;Place le pointeur dans la position indiquée.
DISCH   79     ;Copie le caractère "O" au format ASCII dans la
               ;mémoire-tampon intermédiaire.
DISCH   75     ;Copie le caractère "K" au format ASCII dans la
               ;mémoire-tampon intermédiaire.
COM     0      ;Affiche le contenu de la mémoire-tampon
               ;intermédiaire sur le LCD.
CLEAR          ;Efface la mémoire-tampon intermédiaire et
               ;place le pointeur sur la première position.
DATE           ;Copie la date dans la mémoire-tampon intermédiaire.
DISCH   32     ;Copie le caractère "espace en blanc" au format
               ;ASCII dans la mémoire-tampon intermédiaire.
TIME           ;Copie l'heure dans la mémoire-tampon intermédiaire.
COM     1      ;Transmet le contenu de la mémoire-
               ;tampon intermédiaire via le port de comm.
```

Résultat :

H	H	:	M	M	:	S	S							O	K
J	J	/	M	M	/	A	A								

(Afficheur LCD)

JJ/MM/AA HH:MM:SS

(Transmission via COM1)

B.- Activation d'une sortie numérique le 23 avril 1999 à 11h30 pendant 10 secondes.

```
        RESET   400     ;Désactive le relais 400.
        MOVRI   47      ;Charge le registre de l'année en cours.
        MOVCI   99      ;Charge la constante 99 dans la pile
                        ;arithmétique.
        CPEI    eti1    ;Détecte si l'année coïncide avec celle
                        ;de l'activation de la sortie numérique.
        JMP     exit    ;Saute à "exit" si la comparaison est
                        ;infructueuse.
eti1    MOVRI   46      ;Charge le registre du mois en cours.
        MOVCI   4       ;Charge la constante 4 dans la pile
                        ;arithmétique.
        CPEI    eti2    ;Détecte si le mois coïncide avec celui de
                        ;l'activation de la sortie numérique.
        JMP     exit    ;Saute à "exit" si la comparaison est
                        ;infructueuse.
eti2    MOVRI   44      ;Charge le registre du jour en cours.
        MOVCI   23      ;Charge la constante 23 dans la pile
                        ;arithmétique.
        CPEI    eti3    ;Détecte si le jour coïncide avec celui de
                        ;l'activation de la sortie numérique.
        JMP     exit    ;Saute à "exit" si la comparaison est
                        ;infructueuse.
eti3    MOVRI   43      ;Charge le registre des minutes (nombre
                        ;de minutes écoulées dans la journée).
        MOVCI   690     ;Charge l'équivalent de 11h30 en minutes
                        ;(11;* 60 + 30 = 690).
        CPEI    eti3    ;Détecte si l'heure coïndice avec celle de
                        ;l'activation de la sortie numérique.
        JMP     exit    ;Saute à "exit" si la comparaison est
                        ;infructueuse.
eti3    SET     400     ;Active le relais interne qui indique que la
                        ;sortie numérique doit être activée.
exit    LD      400     ;Début maille de contrôle.
        OR      101
        ANDNT   401
        ANDNT   402
```

```
        OUT     101             ;Maille de contrôle de la sortie numérique.
        LD      101
        TIM     250     100     ;Temporisation de 10 secondes à compter
                                ;de l'activation du relais 400.
        OUT     401             ;Relais interne de désactivation de la sortie
                                ;numérique (se déclenche au bout de 10
                                ;secondes).
        LD      401
        AND     101
        OR      402
        AND     400
        OUT     402             ;Relais empêchant que la sortie numérique
                                ;se réactive au cours de la minute pendant
                                ;laquelle l'état du relais 400 est "1".
        ...                     ;Poursuite de l'exécution du programme.
```

Le relais interne 402 s'emploie pour empêcher que la sortie numérique se réactive lorsque les 10 secondes d'activation sont écoulées puisque l'état du relais 400 reste "1" tant que le programme détecte que la comparaison de la ligne "éti3" indique que les deux valeurs sont ÉGALES (1 minute).

Cet exemple n'est valable que si le MIDA 64 est en marche à la date à laquelle la sortie numérique doit être connectée et s'il est inutile de prévoir qu'il peut se trouver à l'arrêt.

C.- Activation d'une sortie numérique tous les mercredis à 12h00, pendant une minute.

```
        MOVRI   45      ;Charge le jour de la semaine en cours.
        MOVCI   2       ;Charge la constante 2 dans la pile arithmétique.
        CPEI    eti1    ;Détecte si le jour en cours est un mercredi (2).
        JMP     eti2    ;Saute à "éti2" si la comparaison est infructueuse.
eti1    SET     400     ;Active le relais interne qui indique l'activation de
                        ;la sortie numérique.
        LD      400     ;Charge l'état du relais 400 dans la pile logique.
        OUT     101     ;Contrôle de la sortie numérique.
```

```
        JMP     fin     ;Saute à la ligne "fin".
eti2    RESET   400     ;Désactive le relais.
fin     END
```

D.- Stockage de la date, de l'heure et de la valeur d'une entrée analogique dans une table (array) de registres chaque fois qu'une entrée numérique est activée.

```
        LD      0       ;Charge l'état de l'entrée numérique 0 dans
                        ;la pile logique.
        ANDNT   400
        OUT     401     ;Détecte le flanc d'activation de l'entrée
                        ;numérique.
        LD      0       ;Charge l'état de l'entrée numérique 0 dans
                        ;la pile logique.
        OUT     400     ;Stocke l'état de l'entrée sur le relais
                        ;interne de mémoire de l'état précédent.
        LD      400     ;Charge l'état du relais 400 dans la pile
                        ;logique.

        JZ      eti1    ;Détecte si les données doivent être
                        ;stockées dans la table des registres.

                ;À partir de là, les données sont stockées dans la table
                ;des registres.

        MOVRI   60      ;Charge l'entrée analogique dans la pile
                        ;arithmétique.
        STOI    300     ;Stocke la donnée de l'entrée analogique
                        ;dans le registre entier 300 pour fixer sa
                        ;valeur instantanée.
        MOVRI   101     ;Charge le jour du mois dans la pile
                        ;arithmétique.
        STOIX   point   ;Stocke le jour du mois dans le registre
                        ;entier signalé par "point".
        INC     point 1 ;Incrémente la valeur du registre pointeur
                        ;dans la table des registres.
        MOVRI   46      ;Charge le mois de l'année dans la pile
                        ;arithmétique.
```

```
        STOIX   point       ;Stocke le mois de l'année dans le registre
                            ;signalé par le pointeur.
        INC     point  1    ;Incrémente la valeur du registre pointeur
                            ;dans la table des registres.
        MOVRI   103         ;Charge l'année dans la pile arithmétique.
        STOIX   point       ;Stocke le mois de l'année dans le registre
                            ;signalé par le pointeur.
        INC     point  1    ;Incrémente la valeur du registre pointeur
                            ;dans la table des registres.
        MOVRI   99          ;Charge, dans la pile arithmétique, le
                            ;nombre de minutes qui se sont écoulées
                            ;depuis la dernière heure zéro.
        STOIX   point       ;Stocke le jour du mois dans le registre
                            ;entier signalé par "point".
        INC     point  1    ;Incrémente la valeur du registre pointeur
                            ;dans la table des registres.
        MOVRI   300         ;Charge le registre entier 300 dans la pile
                            ;arithmétique.
        STOIX   point       ;Stocke la valeur de l'entrée analogique au
                            ;moment de l'activation de l'entrée
                            ;numérique dans le registre entier signalé
                            ;par "point".
        INC     point  1    ;Incrémente la valeur du registre pointeur
                            ;dans la table des registres pour qu'il soit
                            ;prêt pour le prochain cycle d'écriture dans
                            ;la table des registres.
eti1    ...                 ;Poursuite de l'exécution du programme.
```

E.- Modification de l'heure/la date au moyen de l'instruction CLKP.
La sortie 104 est programmée en tant que sortie des impulsions de manière à ce qu'elle ne s'arrête pas et à ce que l'exécution du programme se poursuive lors de la modification des paramètres mentionnés.

```
;Définition des libellés
texte0      lite    "APPUYEZ SUR F1"

ini     LD      1           ;Charge l'état de l'entrée 1 dans la pile
                            ;logique.
```

```
OR      400         ;Effectue un OR logique entre le relais 400
                    ;et l'entrée 1.
ANDNT   2           ;Effectue un ANDNT avec l'entrée 2.
OUT     400         ;Décharge le résultat du bloc logique sur le
                    ;relais 400.
LD      400         ;Charge l'état du relais 400 dans la pile
                    ;logique.
ANDNT   401         ;Effectue un ANDNT avec le relais 401.
TIM     250   10    ;Charge le temporisateur.
OUT     105         ;Décharge le résultat sur la sortie 105.
LD      105         ;Charge l'entrée 105 dans la pile logique.
TIM     251   10    ;Charge le temporisateur.
OUT     401         ;Décharge le résultat du bloc logique sur le
                    ;relais 401.
CLEAR               ;Efface la mémoire-tampon intermédiaire et
                    ;place le pointeur sur la première position.
DISL    texte0      ;Copie, dans la mémoire-tampon
                    ;intermédiaire, le message de la table de
                    ;libellés "texte0".
LOC     17          ;Place le pointeur dans la position
                    ;indiquée.
TIME                ;Copie l'heure/la date dans la mémoire-
                    ;tampon intermédiaire.
COM     0           ;Affiche le contenu de la mémoire-tampon
                    ;intermédiaire sur le LCD.
RESET   104         ;Désactive le relais 104
INK     356         ;Détection d'une touche.
JZ      ini         ;Si la touche n'a pas été activée, le
                    ;programme saute à "ini".
CLEAR               ;Efface la mémoire-tampon intermédiaire et
                    ;place le pointeur sur la première position.
CLKP                ;Permet d'introduire l'heure/la date.
COM     0           ;Affiche le contenu de la mémoire-tampon
                    ;intermédiaire sur le LCD.
LD      391         ;Charge l'état du relais "Relais entrée"
                    ;dans la pile logique.
OUT     104         ;Décharge l'état de la pile logique sur la
                    ;sortie 104.
END
```

Comment structurer un programme ? C'est la question que nous nous posons tous lorsque nous devons en élaborer un. L'objectif de ce paragraphe et de ce manuel n'est pas de donner un cours de programmation, mais plutôt de vous fournir quelques suggestions destinées, au minimum, à vous aider à vous organiser pour mener cette tâche à bien.

Bien que nous sachions que cela est évident, nous tenons à vous rappeler ce que nous voulons dire lorsque nous parlons de programme. Il s'agit d'une séquence d'instructions qui seront exécutées par un ordinateur pour obtenir un résultat donné.

En ce qui concerne les équipements MIDA, les programmes peuvent être structurés de la manière suivante :

- ***Programme principal.***
- ***Procédures principales.***
- ***Sous-routines secondaires.***

Bien que cela ne soit pas strictement nécessaire, nous vous recommandons de structurer tous les programmes MIDA de cette manière.

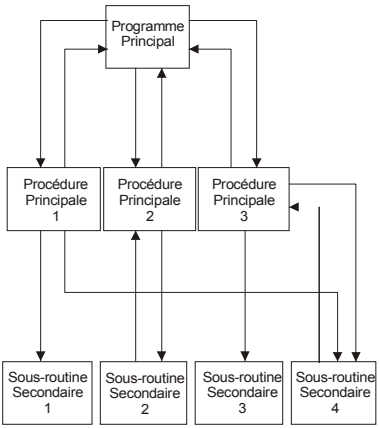
Nous analysons ci-dessous chacun des éléments qui composent cette structure :

- ***Programme principal*** : il s'agit de la partie du programme depuis laquelle toutes les procédures principales (commande des automatismes, affichage du programme, reconnaissance du clavier, détection des alarmes, etc.) sont appelées. Il doit se terminer par l'instruction END pour marquer la fin d'un cycle ou d'un programme.

- ***Procédures principales*** : il s'agit de sous-routines qui sont appelées depuis le programme principal au moyen de l'instruction CALL et qui définissent les différentes tâches qui seront effectuées par le programme de l'équipement. Elles doivent se terminer par l'instruction RET qui entraîne un retour au programme principal.

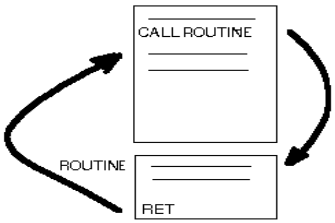
- ***Sous-routines secondaires*** : il s'agit de sous-routines pouvant être appelées depuis n'importe quelle procédure principale et autant de fois que cela s'avère nécessaire au moyen de l'instruction CALL pour réaliser des tâches accessoires. Elles doivent se terminer par l'instruction RET qui entraîne un retour à la procédure principale correspondante.

Vous trouverez, ci-après, un schéma représentant un exemple de structure de programme :

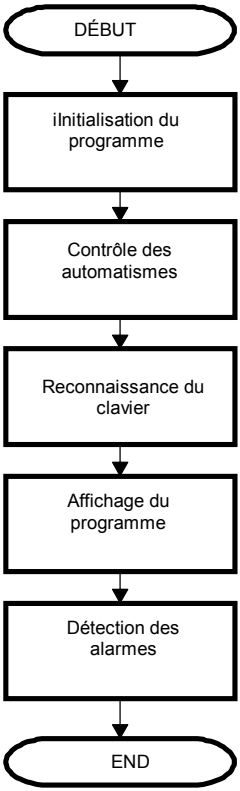


Arrivés à ce point, il est logique de se poser la question : Qu'est-ce qu'une sous-routine ?

Une sous-routine est un ensemble d'instructions qui peuvent être appelées depuis un programme ou une procédure principale au moyen de l'instruction **CALL**. L'instruction **RET** permet d'en sortir et de retourner au point depuis lesquelles elles ont été appelées.



Vous trouverez, ci-après, le diagramme de flux d'un programme uniquement développé jusqu'au stade des procédures principales :



Le code de programme associé à ce diagramme de flux est le suivant :

```
;PROGRAMME PRINCIPAL
        CALL    DEBUT
        CALL    MOVE
        CALL    CLAVIER
        CALL    AFFICH.
        CALL    ALARME
        END
```

Chacune des lignes de ce programme est une procédure principale où sont effectuées des tâches principales qui, à leur tour, appellent des sous-routines secondaires.
Cette façon de faire facilite l'introduction d'éventuelles modifications dans le diagramme car il suffit de modifier la procédure principale affectée. Chaque procédure principale peut comporter autant d'appels de sous-routines secondaires que cela s'avère nécessaire.

REMARQUE : La pile de sous-routines pouvant stocker au maximum 50 adresses de retour, il convient de veiller à ne pas avoir 51 appels (CALL) ouverts.

Vous trouverez, ci-après, une description des instructions de ce groupe :

CALL	**RET**	
JMP	**NOP**	**END**

CALL

CALL XXXX

MNÉMONIQUE : **CALL**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **95**

DESCRIPTION :

APPEL d'une sous-routine.
L'opérande est le numéro ou l'étiquette de la ligne de programme où débute la sous-routine.
Incrémente la pile de sous-routines de 1 niveau (+1).

☑ Exemple 1 :

```
        CALL    100     ;Arrivé à cette ligne, le programme
                        ;saute à la ligne 100 où débute la
                        ;sous-routine correspondante.
```

☑ Exemple 2 :

```
REMO    equ     200

        CALL    REMO    ;Arrivé à cette ligne, le programme
                        ;saute à la ligne 200, signalée par
                        ;l'étiquette et où débute la sous-routine
                        ;correspondante.
```

RET

MNÉMONIQUE : **RET**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **96**

DESCRIPTION :

RETOUR depuis une sous-routine.
Le retour s'exécute au moyen de l'instruction CALL, qui renvoie à la ligne qui suit celle où l'appel a été effectué.
Décrémente la pile de sous-routines de 1 niveau (-1).

☑ Exemple :

```
RET         ;Arrivé à cette ligne, le programme saute à la ligne qui
            ;suit celle où l'exécution de l'instruction CALL
            ;correspondante a eu lieu.
```

JMP

JMP XXXX

MNÉMONIQUE : **JMP**
OPÉRANDE XXXX : **Numéro ou étiquette de ligne de programme**
CODE INSTRUCTION : **92**

DESCRIPTION :

SAUT **inconditionnel** à une ligne de programme.
L'opérande est le numéro ou l'étiquette de la ligne de destination du saut.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
JMP     3000    ;Arrivé à cette ligne, le programme saute à
                ;la ligne 3000. Son exécution se poursuit à
                ;partir de cette ligne.
```

NOP

MNÉMONIQUE : **NOP**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **0**

DESCRIPTION :

SANS EFFET.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
LD    100
OUT   300
NOP         ;Cette ligne de programme n'a aucun effet.
END
```

END

END

MNÉMONIQUE : **END**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **98**

DESCRIPTION :

Indique la FIN du programme.
Le programme retourne à la ligne zéro, l'équipement détecte les erreurs de durée d'exécution et les relais internes appropriés sont activés.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
END     ;Arrivé à cette ligne, le programme retourne à la ligne 0
        ;et l'équipement recherche d'éventuelles erreurs de
        ;durée d'exécution.
```

REMARQUE : Tous les programmes doivent impérativement exécuter l'instruction END pour que l'équipement soit capable de détecter d'éventuelles erreurs de durée d'exécution du programme MIDA.

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Élaboration du squelette structuré d'un programme.

```
;DÉFINITION DES ÉTIQUETTES
;
expl   equ    xxx     ;xxx = relais 1ère exploration

;PROGRAMME PRINCIPAL

temp   CALL debut              ;Initialisation du programme.
       CALL temp               ;Lecture de la température.
       CALL correct.           ;Correction du résultat en degrés.
       CALL clavier            ;Détection du clavier.
       CALL affich             ;Affichage du programme.
       END

;PROCÉDURES PRINCIPALES
       ;INITIALISATION DU PROGRAMME
debut          LD     expl     ;Charge, dans la pile logique, l'état du
                               ;relais lors de la 1ère exploration
                               ;(uniquement à "0" lors de la première
                               ;exploration du programme).
               JZ     deb0     ;Détecte le relais de première exploration
                               ;de programme en vue d'une initialisation
                               ;préalable des données.
               RET
               ...     ;C'est à ce moment qu'ont lieu toutes les opérations
               ...     ;préalables à l'exécution du programme et qui doivent
               ...     ;être exécutées une seule fois, lors de la mise sous
               ...     ;tension de l'équipement.
deb0           ...
               ...
               RET
       ;LECTURE DE LA TEMPÉRATURE
temp           ...     ;Lecture de la sonde de température,
               ...     ;calcul et résultat final en degrés.
```

```
                ...
                RET

;CORRECTION DU RÉSULTAT EN DEGRÉS
correct.        ...     ;Correction de la température mesurée en degrés
                ...     ;en fonction de conditions particulières.
                ...
                RET

;RECONNAISSANCE DU CLAVIER
clavier         ...     ;Sous-routine permettant d'élaborer un menu où il est
                ...     ;possible de détecter la touche activée pour sauter à
                ...     ;différentes sous-routines secondaires.
                ...
                RET
;AFFICHAGE DU PROGRAMME
affich.         ...     ;Sous-routine permettant d'afficher des données et des
                ...     ;messages.
                ...
                RET
;
;SOUS-ROUTINES SECONDAIRES
;
www             ...     ;Sous-routine secondaire appelée par la sous-routine de
                ...     ;reconnaissance du clavier.
                RET

xxx             ...     ;Sous-routine secondaire appelée par la sous-routine de
                ...     ;reconnaissance du clavier.
                RET
yyy             ...     ;Sous-routine secondaire appelée par la sous-routine de
                ...     ;reconnaissance du clavier.
                RET
zzz             ...     ;Sous-routine secondaire appelée par la sous-routine de
                ...     ;reconnaissance du clavier.
                RET
```

ES

Certains équipements MIDA disposent d'entrées numériques possédant une double caractéristique : elles peuvent être utilisées d'une part en tant qu'entrées numériques proprement dites et d'autre part en tant qu'entrées de comptage rapide par interruption.

Il convient de signaler qu'il n'est nécessaire de modifier aucun paramètre pour utiliser ces entrées dans un mode ou l'autre. Le programme MIDA peut employer ces entrées d'une manière ou de l'autre sans intervention préalable sur les paramètres.

Les équipements disposent donc d'une ou de plusieurs entrées de comptage rapide par interruption. Chacune de ces entrées est associée à une autre entrée pour qu'il soit possible de sélectionner le sens du comptage (consultez le Manuel Utilisateur de l'équipement correspondant). Même si ces entrées de comptage ne sont pas utilisées par le programme MIDA, les registres qui leur sont associés sont mis à jour par interruption.

Ces entrées peuvent être utilisées pour des capteurs d'impulsions, des encodeurs, des détecteurs de proximité, des détecteurs magnétiques, etc.

✓ Propriétés du comptage

Le comptage s'effectue par interruption et ne nuit pas à la durée d'exploration du programme MIDA.

Ces entrées sont du même type que les entrées numériques discrètes de l'équipement : exemptes de tension (par contact) ou pour détecteurs NPN.

Chacune de ces entrées dispose de registres entiers de 16 bits associés qui varient selon le modèle d'équipement MIDA à programmer (consultez le Manuel Utilisateur de l'équipement correspondant).

Sur certains équipements MIDA, les registres entiers associés sont différents si le comptage s'effectue par impulsions ou à l'aide d'un encodeur.

REGISTRE ENTIER	DESCRIPTION
Nombre total d'impulsions	Registre qui dénombre et totalise les impulsions.
Destination des impulsions	Registre dans lequel la présélection du comptage doit être programmée. Chaque fois que l'entrée détecte une impulsion, le système vérifie si le comptage a atteint la valeur présélectionnée pour désactiver le relais correspondant.
Nombre partiel d'impulsions	Registre contenant le comptage partiel c'est-à-dire le nombre d'impulsions dénombrées de zéro à la valeur présélectionnée du registre "Destination des impulsions". Une remise à zéro s'effectue automatiquement chaque fois que la valeur présélectionnée est atteinte.
Différence (destination - nombre partiel)	Registre contenant la différence, en valeur absolue, entre le registre "Destination des impulsions" et le registre "Nombre partiel d'impulsions".
Relais associé	Registre contenant l'adresse du relais qui est désactivé lorsque le registre "Nombre total d'impulsions" et le registre "Nombre partiel d'impulsions" contiennent la même valeur.

✓ **Mode de programmation**

Le traitement des registres ci-dessus est identique à celui des registres entiers décrits précédemment : ils peuvent être consultés, comparés, modifiés, etc. par l'intermédiaire d'un programme ou depuis un PC.

Chaque fois que l'équipement détecte une impulsion, il la comptabilise dans le registre **"Nombre total d'impulsions"** et vérifie si le contenu de ce registre est ÉGAL à celui du registre de fin de comptage (**"Destination des impulsions"**).

Si ces deux registres contiennent la même valeur, l'équipement désactive le relais (relais interne, sortie numérique, etc.) dont l'adresse est stockée dans le registre **"Relais associé".**

S'il n'est pas nécessaire de désactiver un relais, il vous suffit de programmer une présélection impossible, par exemple une valeur négative (-32000), pour que le système n'atteigne jamais la fin du comptage ; vous pouvez aussi programmer une adresse de relais de fin de comptage n'affectant pas le déroulement du programme.

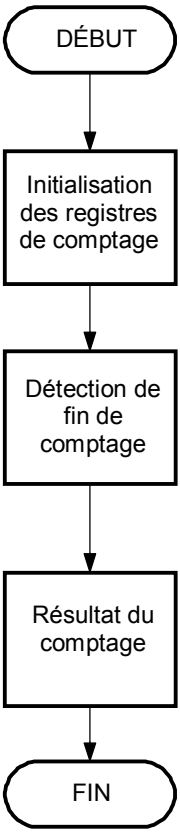
L'écriture d'un programme de comptage comporte les phases suivantes :

- La première phase consiste à **initialiser** le registre "Nombre total d'impulsions", c'est-à-dire à effectuer une remise à zéro ou à stocker la valeur actuelle dans un registre intermédiaire.

- La seconde phase consiste à **programmer** le registre "Destination des impulsions" de comptage à l'aide des données appropriées et à programmer le registre "Relais associé" contenant l'adresse du relais correspondant. Il s'agit généralement de la sortie numérique qui active le mouvement de la machine que vous souhaitez positionner ; l'équipement désactive cette sortie par interruption lorsque les registres "Nombre total d'impulsions " et "Destination des impulsions" contiennent la même valeur.

Mode d'emploi : les entrées de comptage peuvent essentiellement être utilisées à des fins de :

- Positionnement.
- Comptage des impulsions.



- ***Positionnement*** : vous pouvez employer le relais désigné par le registre **"Relais associé"** pour mettre la machine en mouvement. Ce relais sera désactivé par interruption lorsque les registres **"Nombre total d'impulsions"** et **"Destination des impulsions"** contiendront la même valeur.

- ***Comptage des impulsions*** : vous pouvez stocker une valeur impossible à atteindre par le système dans le registre **"Destination des impulsions"** pour collecter, par l'intermédiaire d'un programme, le nombre d'impulsions en vue d'une comparaison ou d'un traitement arithmétique ou utiliser le relais désigné par le registre **"Relais associé"** pour détecter la concordance avec une valeur présélectionnée.

La ***fin de comptage*** se manifeste par la désactivation du relais interne ou de la sortie numérique dont l'adresse est stockée dans le registre **"Relais associé"**.

Si vous souhaitiez ***positionner*** la machine, le relais de fin de comptage peut avoir désactivé le mouvement.

Si votre intention était de ***compter les impulsions*** et si vous n'avez pas remis le registre **"Nombre total d'impulsions"** à zéro, il vous faut déduire la valeur initiale lue en début de comptage de celle du registre **"Nombre total d'impulsions"**.

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Cet exemple consiste à positionner une machine à 100 cm (1.000 mm) de son point d'origine et à afficher l'utilisation de tous les relais associés à l'entrée de comptage rapide, qui est ici l'entrée 0.

```
        LD      399             ;Charge le relais de première exploration
                                ;dans la pile logique.
        JNZ     ini             ;Si l'état du relais est "1", le programme
                                ;saute à "ini".
        SETRI   0       0       ;Stocke la constante 0 dans le registre
                                ;entier 0 (mise à zéro du registre "Nombre
                                ;total d'impulsions").
        SETRI   1       100     ;Stocke la constante 100 dans le registre
                                ;entier 1 (fin du positionnement dans le
                                ;registre "Destination des impulsions).
        SETRI   2       101     ;Stocke la constante 101 dans le registre
                                ;entier 2 (assignation du "Relais associé").
ini     SET     101             ;Active la sortie 101("Relais associé").
        MOVRI   0               ;Charge le contenu du registre entier 0
                                ;(registre "Nombre total d'impulsions) dans
                                ;la pile arithmétique.
        MOVRI   1               ;Charge le registre entier 1 (registre
                                ;"Destination des impulsions") dans la pile
                                ;arithmétique.
        CPGEI   aff             ;Compare si le contenu du registre entier 0
                                ;est supérieur ou égal à celui du registre
                                ;entier 1. Saute à "aff".
        JMP     fin             ;Saut inconditionnel à "fin".
aff     INK     356             ;Détection d'une touche.
        JZ      aff             ;Si la touche n'a pas été activée, le
                                ;programme saute à "aff".
        SETRI   0       0       ;Si la touche a été activée, stocke la
                                ;constante 0 dans le registre entier 0
                                ;(remise à zéro du registre "Nombre total
                                ;d'impulsions").
fin     END                     ;Fin du programme.
```

Les équipements MIDA peuvent disposer d'une entrée d'interruption software.
Il s'agit généralement d'une entrée numérique qui, étant donné ses caractéristiques de fonctionnement, peut se comporter de trois manières différentes :

- Elle peut être utilisée comme une entrée numérique proprement dite.
- Elle peut être utilisée comme une entrée de comptage rapide par interruption.
- Elle peut être utilisée comme une entrée d'interruption software.

L'entrée possédant ces caractéristiques de fonctionnement est indiquée dans le Manuel Utilisateur.

Une interruption software est une sous-routine dont l'exécution est liée à l'activation d'une entrée numérique de l'équipement et qui interrompt l'exécution normale du programme.
Cela signifie donc que le programme n'actualise ni les compteurs, ni les temporisateurs, ni les bases de temps, ni les entrées, etc. pendant toute la durée de l'exécution de l'interruption software.
Cette sous-routine étant terminée, l'exécution du programme reprend sur la ligne où s'est produite l'interruption.

Une “sous-routine” d'interruption software doit être identifiée de manière particulière, grâce à des *pseudo-instructions*, pour que le compilateur la localise parmi les lignes de programme indiquées à cette fin.
Les pseudo-instructions de compilation associées à l'usage des interruptions software sont :

INTER	**END_INTER**

et l'instruction associée est :

IRET

Si un programme ne dispose pas des pseudo-instructions de compilation mentionnées, l'entrée correspondante ne se comporte pas comme une entrée d'interruption logicielle. Elle devient une simple entrée de comptage, une entrée numérique discrète.

✓ Mode de programmation

À titre d'exemple de traitement d'une interruption, nous avons choisi d'écrire le programme d'un compteur d'impulsions tenant compte du sens.

```
        INTER                ;Début de l'interruption software.
        LD        1          ;Charge l'état de l'entrée 1 dans la pile
                             ;logique.
        JZ        eti1       ;Détermine si le registre entier "compt"
                             ;doit être incrémenté ou décrémenté.
        INC       compt -1   ;Décrémente le registre entier "compt".
        IRET                 ;Retour de l'interruption au programme
                             ;principal du MIDA.
eti1    INC       compt 1    ;Incrémente le registre "compt".
        IRET                 ;Retour de l'interruption au programme
                             ;principal du MIDA.
        END_INTER            ;Fin de l'interruption software.
```

L'exemple ci-dessus vous permettra de déduire que :

- Les *pseudo-instructions de compilation* INTER et END_INTER se placent respectivement au début et à la fin du code du programme d'interruption.
  Le code du programme d'interruption peut donc être placé à n'importe quel endroit du programme MIDA.
- L'instruction à l'origine du *retour* au programme principal MIDA est IRET. Elle doit être insérée à la fin du programme d'interruption (souvenez-vous que la fin du code du programme d'interruption est indiquée par la pseudo-instruction END_INTER).
  Il est donc impossible d'insérer une instruction RET à l'intérieur d'une interruption : le compilateur vous retournera un message d'erreur.

- L'emploi de toutes les autres instructions du jeu d'instructions de l'équipement est *autorisé* à l'intérieur de cette sous-routine.
- Les *instructions de saut* (JZ, JNZ, JMP) ne peuvent être employées pour sauter de l'extérieur à l'intérieur de l'interruption. Tout saut à l'extérieur d'une interruption entraîne l'impossibilité de retourner à l'intérieur de celle-ci.
- Il est néanmoins possible d'*appeler des sous-routines* grâce à l'instruction CALL, depuis le module d'interruptions.

Important :
Veillez à ce que la *durée d'exécution* des instructions contenues dans l'interruption (sous-routines comprises) soit inférieure au temps nécessaire à l'exécution d'importants processus du programme MIDA tels que la lecture des entrées, le comptage, les temporisations, etc. pour éviter que ceux-ci soient interrompus ou ignorés.

✓ **Description**

PSEUDO-INSTRUCTION DE COMPILATION	DESCRIPTION
INTER	Pseudo-instruction de compilation indiquant au compilateur où commence l'interruption.
END_INTER	Pseudo-instruction de compilation indiquant au compilateur où s'achève l'interruption.

IRET

MNÉMONIQUE : **IRET**
OPÉRANDE XXXX : **Aucun**
CODE INSTRUCTION : **91**

DESCRIPTION :

Instruction à l'origine du saut de retour de l'interruption software à l'exécution normale du programme.
Le retour renvoie à la ligne qui suit celle où a eu lieu la détection de l'interruption, par l'intermédiaire de l'entrée numérique correspondante.
Ne fait pas intervenir les piles (0).

☑ Exemple :

```
        INTER                ;Début de l'interruption software.
        LD      1            ;Charge l'état de l'entrée 1 dans la pile
                             ;logique.
        JZ      eti1         ;Détermine si le registre entier "compt"
                             ;doit être incrémenté ou décrémenté.
        INC     compt -1     ;Décrémente le registre entier "compt".
        IRET                 ;Retour de l'interruption au programme
                             ;principal du MIDA où a eu lieu la détection
                             ;de celle-ci.
eti1    INC     compt 1      ;Incrémente le registre "compt".
        IRET                 ;Retour de l'interruption au programme
                             ;principal du MIDA où a eu lieu la détection
                             ;de celle-ci.
        END_INTER            ;Fin de l'interruption software.
```

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Utilisation de l'entrée numérique correspondant à celle des interruptions logicielles pour détecter des alarmes et les stocker dans une table de registres.

```
;Programme principal MIDA
        LD          10          ;Charge l'état de l'entrée 10 dans la pile
                                ;logique.
        OR          101
        ANDNT       11
        OUT         101         ;Commande marche/arrêt de la sortie
                                ;numérique 101.
        ...
;Fin du programme

;Sous-routine de suivi de la table des registres, (elle peut être appelée
;depuis n'importe quel point du programme).
ant     MOVRI       point_texte     ;Charge le contenu du registre
                                    ;entier dans la pile arithmétique.
        STOIX       point_table     ;Stocke, dans une table de
                                    ;registres, l'adresse du texte
                                    ;associé à l'entrée numérique
                                    ;d'alarme qui a été activée.
        INC         point_table 1   ;Incrémente le pointeur dans la
                                    ;table de registres en vue de la
                                    ;prochaine inscription dans la table.
        RET

;Début de l'interruption software. Cet ensemble d'instructions est
;exécuté chaque fois que l'entrée est activée.
        INTER                       ;Début de l'interruption.
        SETRI       point_texte 0   ;Initialise le pointeur dans la table
                                    ;des libellés.
        LD          12              ;Charge l'état de l'entrée numérique
                                    ;12 dans la pile logique.
```

```
        JZ          eti1                ;Saute à "éti1" si l'alarme n'a pas
                                        ;été activée.
        CALL        ant                 ;Stocke dans la table des registres
                                        ;si l'entrée 12 est activée.
eti1    INC         point_texte  1      ;Incrémente le pointeur dans la
                                        ;table des libellés.
        LD          13                  ;Charge l'état de l'entrée numérique
                                        ;13 dans la pile logique.
        JZ          eti2                ;Saute à "éti2" si l'alarme n'a pas
                                        ;été activée.
        CALL        ant                 ;Stocke dans la table des registres
                                        ;si l'entrée 13 est activée.
eti2    INC         point_texte  1      ;Incrémente le pointeur dans la
                                        ;table des libellés.
        LD          14                  ;Charge l'état de l'entrée 14 dans la
                                        ;pile logique.
        JZ          eti3                ;Saute à "éti3" si l'alarme n'a pas
                                        ;été activée.
        CALL        ant                 ;Stocke dans la table des registres
                                        ;si l'entrée 14 est activée.
eti3    INC         point_texte  1      ;Incrémente le pointeur dans la
                                        ;table des libellés.
        IRET                            ;Retour de l'interruption.
        END_INTER                       ;Fin de l'interruption software.
;
;Fin de l'interruption software.
```

Il n'est pas recommandé d'utiliser des sous-routines à l'intérieur d'une interruption en raison de la durée d'exécution.
Veillez particulièrement à ce que la durée TOTALE (sous-routines comprises) d'exécution de l'interruption soit inférieure au temps qui sépare deux interruptions.

B.- Utilisation de l'entrée numérique correspondant à celle des interruptions software pour compter les événements et détecter si le système atteint une valeur présélectionnée stockée dans un registre entier normal.

```
        INTER                   ;Début de l'interruption.
        SET     relais_fin      ;Active le relais interne qui indique
                                ;la fin du comptage.
        INC     compt   1       ;Incrémente le compteur d'événements.
        MOVRI   compt           ;Charge le contenu du registre entier
                                ;"compt" (compteur) dans la pile
                                ;arithmétique.
        MOVRI   presel          ;Charge le contenu du registre entier
                                ;"présél" (présélection) dans la pile
                                ;arithmétique.
        CPGEI   eti1            ;Détecte si le compteur a atteint la valeur
                                ;présélectionnée.
        RESET   relais_fin      ;Désactive le relais interne qui indique la
                                ;fin du comptage.
eti1    IRET                    ;Retour de l'interruption.
        END_INTER               ;Fin de l'interruption software
```

Cet ensemble d'instructions est exécuté chaque fois que l'entrée numérique correspondant à celle des interruptions software est activée.
L'interruption se limite à activer ou désactiver le relais interne et à indiquer si le compteur a atteint ou non la valeur présélectionnée contenue dans un registre entier normal.

C.- Utilisation de l'entrée numérique correspondant à celle des interruptions software pour compter les impulsions par rapport à une base de temps donnée.

```
;Programme principal
        LDNT    400
        TIM     250   10        ;Temporisateur base de temps 1 seconde.
        OUT     400
```

```
        ...                             ;Poursuite de l'exécution du programme.
;Début de l'interruption.
        INTER                           ;Début de l'interruption.
        INC     compt   1               ;Incrémente le compteur d'impulsions.
        LD      250
        JZ      eti1                    ;Détecte si le temps spécifié par la base de
                                        ;temps s'est écoulé.
        MOVRI   compt
        STOI    inter                   ;Stocke le résultat du comptage, dans un
                                        ;registre entier "inter" en vue de calculs
                                        ;ultérieurs et pour gagner du temps à
                                        ;l'intérieur de l'interruption.
eti1    IRET                            ;Retour de l'interruption.
        END_INTER                       ;Fin de l'interruption software.
```

Il convient de se contenter d'effectuer les opérations strictement nécessaires à l'intérieur des interruptions afin de gagner en temps d'exécution.

Dans l'exemple ci-dessus, le résultat du comptage est déchargé dans un registre intermédiaire : il peut ainsi être utilisé ultérieurement dans le programme normal pour effectuer des calculs (vitesse, etc.).
Si au lieu d'utiliser l'instruction TIM, vous employez TIMR, la base de temps se présentera comme suit :

```
        INC     compt   1               ;Incrémente le compteur d'événements.
        MOVRI   compt
        MOVRI   presel
        CPGEI   eti1                    ;Détecte si le compteur a atteint la valeur
                                        ;présélectionnée.
        RESET   relais_fin              ;Désactive le relais interne qui indique la
                                        ;fin du comptage.
eti1    IRET                            ;Retour de l'interruption.
        END_INTER
```

L'interruption se limite à activer ou désactiver le relais interne et à indiquer si le compteur a atteint ou non la valeur présélectionnée contenue dans un registre entier normal.

Les équipements MIDA disposent d'entrées et/ou de sorties analogiques intégrées aux équipements eux-mêmes ou à des modules ou des cartes d'extension.

✓ ENTRÉES ANALOGIQUES

Les entrées analogiques peuvent être de deux types :

- **Entrées analogiques différentielles.**
- **Entrées analogiques courantes.**

Mode de programmation

L'acquisition des valeurs des entrées analogiques est effectuée par l'équipement, par interruption.

Pour obtenir ces valeurs, il vous suffit de lire le contenu des registres entiers correspondants (consultez le Manuel Utilisateur de l'équipement ou celui du module ou de la carte d'extension).

Ces registres sont accessibles par le programme MIDA et peuvent être soumis à des traitements arithmétiques et à des comparaisons, être affichés, etc. Il s'agit en fait de registres entiers normaux dont le contenu est régulièrement mis à jour par l'équipement en fonction de la valeur de l'entrée analogique correspondante.

Les valeurs maximales et minimales du contenu de ces registres dépendent de la résolution du convertisseur.

La ***mise à jour des registres*** des entrées est fonction du signal appliqué à l'entrée correspondante.
Exemple :

```
MOVRI   60      ;Charge la valeur du registre entier 60,
                ;équivalent à une entrée analogique, dans
                ;la pile arithmétique.
STOI    500     ;Stocke le contenu du registre entier 500
                ;dans la pile arithmétique.
```

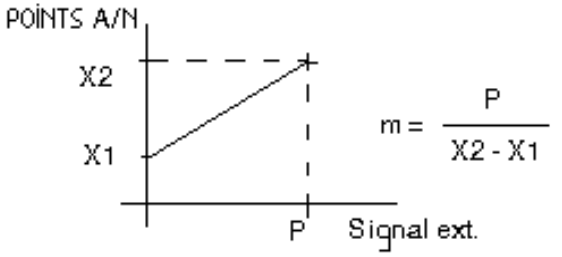
Dans le cas d'une entrée analogique courante, une valeur comprise entre 0 et 4000 sera stockée dans le registre entier 500, alors que dans le cas d'une entrée analogique différentielle, cette valeur, proportionnelle au signal analogique appliqué à l'entrée analogique (registre entier 60), sera comprise entre +/- 32767.

Le ***convertisseur analogique différentiel*** est de type extensiométrique. Il est possible qu'en fin d'échelle, il n'atteigne pas la résolution spécifiée dans le Manuel Utilisateur. Il est donc nécessaire d'effectuer des routines d'étalonnage pour calculer la pente de la droite correspondant à cette entrée.

Exemple :

Supposons que vous souhaitiez obtenir une lecture en deux points, le premier correspondant à une valeur d'entrée zéro et le deuxième à une valeur quelconque, la plus proche possible du maximum de l'échelle du signal externe.

Il convient donc de calculer la pente de la droite à partir des deux points de celle-ci :



Vous obtenez ainsi un facteur d'échelle qui, opéré par le signal extérieur, donne toujours pour résultat un point situé au-dessus de la droite.

En multipliant les points analogiques d'entrée, vous obtiendrez les unités d'ingénierie (kg, ºC, % Hr, etc.), tant pour les entrées de type différentiel que pour les entrées courantes.

Relais Fin de conversion A/N

Certains équipements MIDA possèdent un ou plusieurs relais appelés "Relais Fin de conversion A/N", qui s'activent chaque fois que le convertisseur A/N effectue une lecture. Chaque relais de ce type possède la même adresse que le registre entier correspondant à l'entrée analogique.

Selon le type de convertisseur (c'est-à-dire selon le modèle MIDA ou le modèle d'extension), chaque entrée analogique possède un relais associé ou diverses entrées partagent un même relais (consultez le Manuel Utilisateur de l'équipement MIDA et/ou celui des modules ou de la carte d'extension).

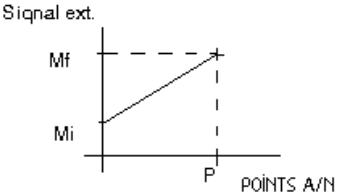
Tenez compte du fait que le "Relais Fin de conversion A/N" s'active la première fois qu'il détecte une conversion A/N et que son état reste "1" tant que vous ne l'avez pas désactivé à l'aide du programme utilisateur. Le relais s'active de nouveau lors de la conversion A/N suivante et peut contrôler toutes les conversions A/N effectuées par le convertisseur.

Comme nous le verrons dans les exemples ci-après, ce relais est très utile pour faire la moyenne des lectures d'une entrée analogique.

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Affichage de la valeur d'une entrée analogique d'un capteur de température, dont les lectures maximale et minimale sont comprises entre 0 et 100ºC.
L'un des exemples correspond à une entrée analogique de 0-20 mA et l'autre à une entrée analogique de 4-20 mA. Dans les deux cas, la résolution est de 400 points.

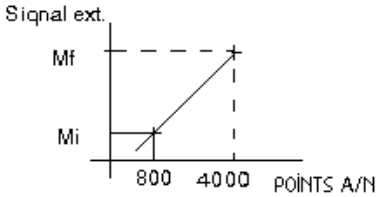


0 - 20 mA

$$T = \frac{Mf - Mi}{4000} * pts\_A/N\_lus + Mi$$

```
MOVIF    ent_1       ;Charge le contenu du reg. entier "ent_1"
                     ;équivalent à une entrée analogique, dans
                     ;un format à virgule flottante.
MOVCF    100         ;Charge la constante à virgule flottante 100
                     ;dans la pile arithmétique.
MULF                 ;Multiplie les données contenues dans la
                     ;pile arithmétique et place le résultat dans
                     ;celle-ci.
MOVCF    4000        ;Charge la constante à virgule flottante
                     ;4000 dans la pile arithmétique.
DIVF                 ;Divise les deux données contenues dans
                     ;la pile arithmétique.
STOF     temper      ;Stocke le résultat précédent dans un
                     ;reg. à virgule flottante, par ex. "tempér".
CLEAR                ;Efface la mémoire-tampon intermédiaire et
                     ;place le pointeur sur la première position.
DISRF    temper  51  ;Charge le contenu du registre à virgule
                     ,flottante "tempér", dans la mémoire-
                     ;tampon intermédiaire, avec une décimale.
```

```
COM     0       ;Affiche le contenu de la mémoire-tampon
                ;intermédiaire sur le LCD.
```



4-20 mA.

$$T = \frac{Mf - Mi}{3200} * (pts\_A/N\_lus - offset) + Mi$$

```
MOVIF   ent_1   ;Charge le contenu du reg. entier "ent_1"
                ;équivalent à une entrée analogique dans
                ;un format à virgule flottante.
MOVCF   800     ;Charge la constante à virgule flottante 800
                ;dans la pile arithmétique.
SUBF            ;Divise les deux données contenues dans
                ;la pile arithmétique (Calcule l'offset du
                ;signal d'entrée de 4 mA).
MOVCF   100     ;Charge la constante à virgule flottante 100
                ;dans la pile arithmétique.
MULF            ;Multiplie les données contenues dans la
                ;pile arithmétique et place le résultat dans
                ;celle-ci.
MOVCF   4000    ;Charge la constante à virgule flottante
                ;4000 dans la pile arithmétique.
DIVF            ;Divise les deux données contenues dans
                ;la pile arithmétique.
STOF    temper  ;Stocke le résultat précédent dans un
                ;reg. à virgule flottante, par ex. "tempér.".
CLEAR           ;Efface la mémoire-tampon intermédiaire et
                ;place le pointeur sur la première position.
DISRF   temper  51  ;Charge le contenu du registre à virgule
                ,flottante "tempér.", dans la mémoire-
                ;tampon intermédiaire, avec une décimale.
COM     0       ;Affiche le contenu de la mémoire-tampon
                ;intermédiaire sur le LCD.
```

B.- Calcul de la moyenne de la valeur d'une entrée analogique à un nombre de reprises fixé.

```
;Définition des étiquettes
fin_an      equ   xxx      ;"Relais Fin de conversion" de l'entrée
                           ;analogique.
ent_1       equ   xxx      ;Registre entier d'une entrée analogique.
compt       equ   xxx      ;Registre entier (comptage des lectures
                           ;effectuées).
result      equ   xxx      ;Registre entier (moyenne des mesures)
inter       equ   xxx      ;Registre à virgule flottante (accumulateur
                           ;des mesures).
moy         equ   xxx      ;Constante entière (nbre de moyennes).
;
        MOVCF     0        ;Charge la constante à virgule flottante 0
                           ;dans la pile arithmétique.
        STOF      inter    ;Initialise un registre à virgule flottante en
                           ;tant qu'accumulateur de mesures et
                           ;stocke la constante 0.
        MOVCI     0        ;Charge la constante indiquée (0) dans la
                           ;pile arithmétique.
        STOI      compt    ;Initialise un registre entier en tant que
                           ;compteur des lectures effectuées et
                           ;stocke la constante 0.
        LD        fin_an   ;Charge, dans la pile logique, l'état du
                           ;relais Fin de conversion "fin_an"
                           ;correspondant à l'entrée analogique.
        JZ        eti      ;Si l'état de ce relais = "1", cela signifie
                           ;qu'une nouvelle lecture analogique a eu
                           ;lieu. Dans le cas contraire, le programme
                           ;saute à la ligne "éti".
        RESET     fin_an   ;Désactivation du relais Fin de conversion
                           ;"fin_an" pour préparer la prochaine
                           ;détection de lecture.
        MOVIF     ent_1    ;Charge, dans la pile, la valeur de la
                           ;lecture analogique convertie au format à
                           ;virgule flottante.
        MOVRF     inter    ;Charge le contenu du registre à virgule
                           ;flottante "inter" dans la pile arithmétique.
```

```
        ADDF                    ;Fait la somme des données de la pile
                                ;arithmétique.
        STOF    inter           ;Accumule les valeurs des lectures
                                ;analogiques.
        INC     compt 1         ;Incrémente d'une unité le compteur de
                                ;lectures.
        MOVRI   compt           ;Charge le contenu du registre entier
                                ;"compt" dans la pile arithmétique.
        MOVCI   moy             ;Charge la constante "moy" (nombre de
                                ;moyennes) dans la pile arithmétique.
        CPLEI   eti             ;Vérifie si le nombre de lectures
                                ;présélectionné a été réalisé.
        MOVRF   inter           ;Charge le contenu du registre à virgule
                                ;flottante "inter" dans la pile arithmétique.
        MOVIF   compt           ;Charge le contenu du registre à virgule
                                ;flottante "compt" dans la pile
                                ;arithmétique.
        DIVF                    ;Calcul de la moyenne arithmétique des
                                ;lectures réalisées
                                ;(accumulateur/nbre_de_lectures).
        STOFI   result          ;Stocke le résultat dans le registre entier
                                ;"résult" après l'avoir converti au format
                                ;entier.
eti     --------                ;Poursuite de l'exécution du programme.
```

C.- Programme d'étalonnage et d'affichage d'une sonde PT100. Ce programme contient des routines d'étalonnage de la sonde PT100 et des routines d'affichage de la température lue par celle-ci en degrés.

```
;Définition des relais
r1          equ         60      ;Relais Fin de conversion entrée A/N.
oui         equ         600     ;Indicateur de confirmation de réponse.
touc1       equ         352     ;Touche <9> Étalonnage de la sonde
                                ;PT100
touc2       equ         355     ;Touche <.> Confirmation.
;Définition des registres entiers
pt100       equ         60      ;Entrée analogique de la sonde PT100.
points      equ         500     ;Intermédiaire points A/N.
temp1       equ         501     ;Points de température minimale.
temp2       equ         502     ;Points de température maximale.
compt       equ         503     ;Compteur de lectures A/N.
inter       equ         504     ;Intermédiaire calculs.
offset      equ         505     ;Offset de la sonde PT100.
;Définition des registres à virgule flottante
accum       equ         0       ;Accumulateur de lectures A/N.
tempox      equ         1       ;Température min. d'étalonnage en degrés.
tempoy      equ         2       ;Température max. d'étalonnage en degrés
tempo       equ         4       ;Température calculée en degrés.
facteur     equ         5       ;Pente de la droite de la sonde PT100.
;Définition de la constante
nfois       equ         9       ;Nombre de lectures A/N pour la moyenne.
;Définition des libellés
texte0      lite        "PT100"
texte1      lite        "PT100 temp. min."
texte2      lite        "Temp. min. étal. ?"
texte3      lite        "PT100 temp. max."
texte4      lite        "Temp. max. étal. ?"
texte5      lite        "APPUYEZ ... <.>"
```

```
;PROGRAMME PRINCIPAL
eti     CALL    cal1        ;Calcul de la température.
        CALL    aff         ;Affichage de la température.
        INK     touc1       ;Détection de la touche.
        JZ      eti0        ;Vérifie si la touche d'étalonnage est
                            ;activée.
        CALL    etal        ;Étalonnage de la sonde PT100.
eti0    END

;CALCUL DE LA TEMPÉRATURE DE LA SONDE PT100
cal1    MOVRI   pt100       ;Charge les points d'entrée de la sonde.
        MOVRI   offset      ;Charge l'offset.
        SUBI                ;Déduit l'offset aux points mesurés.
        STOI    inter       ;Stocke le résultat.
        MOVIF   inter       ;Charge le registre intermédiaire.
        MOVRF   facteur     ;Charge le facteur d'étalonnage.
        DIVF                ;Multiplie pour fournir la température.
        STOF    tempo       ;Température en ºC.
        RET

;AFFICHAGE DU PROGRAMME
aff     CLEAR               ;Efface la mémoire-tampon intermédiaire.
        DISL    texte0      ;Texte "PT100".
        LOC     7           ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        DISRF   tempo   42  ;Affichage de la température.
        LOC     13          ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        DISCH   67          ;Charge le caractère "C".
        DISCH   223         ;Charge le caractère "º".
        COM     0           ;Affichage sur le LCD.
        RET

;ÉTALONNAGE PT100 (facteur = tempoy - tempox / temp2 - temp1).
;Étalonnage de la température minimale (calcul de la pente de la droite).
etal    CLEAR               ;Efface la mémoire-tampon intermédiaire.
        DISL    texte2      ;Texte "Temp. min. étal. ?"
        LOC     28          ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
```

```
        DISCH   67          ;Charge le caractère "C".
        DISCH   223         ;Charge le caractère "º".
        COM     0           ;Affichage sur le LCD.
        LOC     21          ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        INF     6           ;Introduction de la température minimale.
        STOF    tempox      ;Stocke l'introduction.
        CLEAR               ;Efface la mémoire-tampon intermédiaire.
        DISL    texte1      ;Texte "PT100 temp. min."
        LOC     6           ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        DISCH   223         ;Charge le caractère "º".
        LOC     16          ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        DISL    texte5      ;Texte "APPUYEZ... <.> "
        COM     0           ;Affichage sur le LCD.
etal1   CALL    sip         ;Vérifie la confirmation.
        JZ      etal1
        CALL    mes0        ;Prend des points de température minimale
        MOVRI   points      ;Charge les points de moyenne.
        MOVRI   points      ;Charge les points de moyenne.
        STOI    temp1       ;Stocke le calcul de la pente minimale.
        STOI    offset      ;Stocke le calcul de l'offset et passe à
                            ;l'étalonnage de la température maximale
                            ;(calcul de la pente de la droite).
        CLEAR               ;Efface la mémoire-tampon intermédiaire.
        DISL    texte4      ;Texte "Temp. max. étal. ?"
        LOC     28          ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        DISCH   67          ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        DISCH   223         ;Charge le caractère "º".
        COM     0           ;Affichage sur le LCD.
        LOC     21          ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        INF     6           ;Introduction de la température maximale.
        STOF    tempoy      ;Stocke l'introduction.
        CLEAR               ;Efface la mémoire-tampon intermédiaire.
        DISL    texte3      ;Texte "PT100 temp. max."
```

```
        LOC     6           ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        DISCH   223         ;Charge le caractère "º".
        LOC     16          ;Position du pointeur dans la mémoire-
                            ;tampon intermédiaire.
        DISL    texte5      ;Texte "APPUYEZ... <.> "
        COM     0           ;Affichage sur le LCD.
etal5   CALL    sip         ;Vérifie la confirmation.
        JZ      etal5
        CALL    tom0        ;Prend des points de température max.
        MOVRI   points      ;Charge les points de moyenne.
        STOI    temp2       ;Stocke le calcul de la pente maximale.
        MOVIF   temp2       ;Charge le calcul de la pente maximale.
        MOVIF   temp1       ;Charge le calcul de la pente minimale.
        SUBF                ;Calcule la différence de pente.
        MOVRF   tempoy      ;Charge la température max. d'étalonnage.
        MOVRF   tempox      ;Charge la température min. d'étalonnage.
        SUBF                ;Calcul la différence de température.
        DIVF                ;Calcul le facteur de la pente.
        STOF    facteur     ;Stocke le facteur de la pente de la droite.
        RET

;PRISE DE POINTS DE CONVERTISSEUR EN FAISANT LA MOYENNE
mes0    SETRI   accum   0   ;Initialise l'accumulateur de lectures.
        SETRI   compt   0   ;Initialise le compteur de moyennes.
mes1    LD      r1          ;Charge le "Relais Fin de conversion A/N".
        JZ      mes1
        RESET   r1          ;Réinitialise le "Relais Fin de conversion
                            ;A/N".
        MOVIF   pt100       ;Charge les points d'entrée de la sonde.
        MOVRF   accum.      ;Charge l'accumulateur de lectures.
        ADDF                ;Fait la somme des points A/N et de
                            ;l'accumulateur.
        STOF    accum.      ;Stocke dans l'accumulateur de lectures.
        INC     compt   1   ;Incrémente le compteur de lectures.
        MOVRI   compt       ;Charge le compteur de lectures.
        MOVCI   nfois       ;Charge la constante des moyennes.
        CPLEI   mes1        ;Vérifie si le nombre de lectures
                            ;présélectionné a été réalisé.
```

```
        MOVRF   accum.      ;Charge l'accumulateur de lectures.
        MOVIF   compt       ;Charge le compteur de lectures.
        DIVF                ;Divise (accumulateur/moyennes).
        STOFI   points      ;Stocke les points de moyenne.
        RET

;DEMANDE DE CONFIRMATION
sip     SET     si          ;Active l'indicateur de confirmation.
        INK     touc2       ;Détection de la touche.
        JNZ     sip1        ;Vérifie la confirmation.
        RESET   oui         ;Efface l'indicateur de confirmation.
sip1    LD      oui         ;Active l'indicateur de confirmation.
        RET
```

✓ SORTIES ANALOGIQUES

Les équipements MIDA disposent de sorties analogiques intégrées à des modules ou à des cartes d'extension.
Vous trouverez les informations correspondantes dans le Manuel Utilisateur de ces modules et/ou cartes d'extension.

Mode de programmation

Les sorties analogiques des équipements MIDA ont une résolution de 4000 points et leur intensité est comprise entre 0/4 et 20 mA.
L'intensité de la sortie analogique est proportionnelle au contenu du registre entier qui lui est associé.
Si le registre entier xxx (registre entier correspondant à une sortie analogique de 0-20 mA) contient la donnée :

4000, la sortie sera de 20 mA
2000, la sortie sera de 10 mA
0, la sortie sera de 0 mA

Les registres des sorties analogiques sont accessibles par le programme MIDA et peuvent être soumis à des traitements arithmétiques ou à des comparaisons, être modifiées, affichés, etc.

L'intensité des sorties analogiques des équipements MIDA est toujours proportionnelle au contenu du registre.

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Cet exemple consiste à prendre la valeur de deux entrées analogiques (de 32767 points), à calculer la moyenne arithmétique et à produire une sortie analogique (de 4000 points) proportionnelle à ce calcul.
Sortie analogique = (Entrée analogique 1) + (Entrée analogique 2) / 2

```
MOVIF   60    ;Charge, dans la pile arithmétique, la valeur d'une
              ;entrée analogique dans un format à virgule
              ;flottante.
MOVIF   61    ;Charge, dans la pile arithmétique, la valeur de la
              ;seconde entrée analogique dans un format à
              ;virgule flottante.
ADDF          ;Fait la somme des deux données contenues dans
              ;la pile arithmétique et place le résultat dans celle-
              ;ci (Ent_A/N1 + Ent_A/N2 = somme).
MOVCF   2     ;Charge la constante 2 dans la pile arithmétique.
DIVF          ;Divise les deux données contenues dans la pile
              ;arithmétique (somme / 2 =
              ;moyenne_arithmétique) et place le résultat dans
              ;cette même pile.
STOF    0     ;Stocke le contenu de la pile arithmétique
              ;(moyenne arithmétique des deux entrées
              ;analogiques) dans le registre à virgule flottante 0.
MOVRF   0     ;Charge le contenu du registre à virgule flottante 0
              ;dans la pile arithmétique.
MOVCF   4000  ;Charge la constante 4000 dans la pile
              ;arithmétique.
MULF          ;Multiplie les deux données de la pile arithmétique
              ;(multiplication de la moyenne par la résolution de
              ;la sortie analogique).
MOVCF   32767 ;Charge la constante à virgule flottante 32767
              ;(division par la résolution évaluée des entrées
              ;analogiques).
DIVF          ;Divise les données de la pile arithmétique et
              ;place le résultat dans celle-ci (calcul de la sortie
              ;analogique proportionnelle à la moyenne
              ;arithmétique des entrées analogiques à l'aide
              ;d'une règle de trois).
STOFI   1000  ;Le résultat (à virgule flottante) contenu dans la
              ;pile arithmétique doit être converti au format
              ;entier pour pouvoir être stocké dans le registre
              ;entier correspondant à la sortie analogique.
```

FI

Certains équipements MIDA disposent d'une nouvelle instruction appelée **FUNC.**
Cette instruction, qui dépend de l'opérande, permet d'appeler, en interne, une fonction donnée : fonction de contrôle PID, fonction de pesage, etc.

La description de l'instruction **FUNC** varie selon les fonctions appelées. D'une manière générique, le format de cette instruction est le suivant :

FUNC

FUNC XXXX YYYY

MNÉMONIQUE : **FUNC**
OPÉRANDE XXXX : **Constante indiquant la fonction à appeler**
OPÉRANDE YYYY : **Constante indiquant un paramètre optionnel de la fonction appelée.**
CODE INSTRUCTION : **90**

DESCRIPTION :

Description de la fonction et de ses opérandes.

Vous trouverez une description des fonctions disponibles dans les chapitres suivants.

Le programme interne de certains équipements MIDA contient des routines de pesage qui incluent les fonctions spécifiques ci-après :

- Set-up utilisateur / avancée des balances.
- Étalonnage des balances.
- Réglage automatique de la tare.

Le système vous permet de disposer d'un maximum de 18 balances. Leur nombre varie selon le modèle d'équipement MIDA (consultez le Manuel Utilisateur).

✓ Fonctions de programmation du pesage

Pour utiliser les fonctions de pesage, il vous faut employer l'instruction FUNC avec quatre opérandes différents.

```
FUNC  7  n    ;Mise à zéro automatique d'une balance.
FUNC  8  0    ;Étalonnage automatique de la balance.
FUNC  9  0    ;Set-up utilisateur de la balance.
FUNC  9  1    ;Set-up avancée de la balance.
```

✓ Paramètres d'une balance

Les paramètres dont vous disposez pour chacune des balances dépendent du type de fonction appelée :

	Options	FUNC 9 0	FUNC 9 1	FUNC 8	FUNC 7
État	ON / OFF	Accessible	Accessible	------	------
Moyennes	1 / 2 / 4 / 8	Accessible	Accessible	------	------
Cadence	1 / 2 / 5	Accessible	Accessible	------	------
Nbre de décimales	0 / 1 / 2 / 3	Accessible	Accessible	------	------
Facteur Échelle	À virgule flottante	------	Accessible	Automatique	------
Zéro	Entier	------	Accessible	Automatique	------
Tare	Entier	------	Accessible	Effacement	Automatique

✓ Registres utilisés par les balances

Les relais/registres utilisés par les fonctions de pesage pour chaque balance et accessibles par l'utilisateur sont :

- Relais de fin de pesage (relais Fin de conversion A/N).
- Registre entier de points de conversion A/N (entrée analogique).
- Registre à virgule flottante (résultat de la pesée).
- Registre entier indiquant le nombre de décimales.

REMARQUE : Le résultat du pesage est le registre à virgule flottante correspondant au numéro de la balance.

✓ Équations de fonctionnement des balances

Le paramètre "nbre_mesures" spécifie le nombre de mesures qui doivent être réalisées par l'entrée analogique A/N pour obtenir un poids moyen. Ce paramètre permet d'éliminer les informations parasites lors du pesage, mais diminue la vitesse de celui-ci.

$$\text{moyenne} = \frac{1}{\text{nbre\_ mesures}} \sum \text{points convertisseur}$$

Il convient tout d'abord de calculer la moyenne des points de convertisseur de l'entrée, puis de soustraire la valeur du zéro, la tare, et de diviser le résultat par le facteur d'échelle.

$$\text{poids} = \frac{\text{moyenne - zéro - tare}}{\text{facteur échelle}}$$

Il suffit ensuite d'éliminer les décimales conformément aux indications fournies par le paramètre "nombre de décimales" et d'arrondir le résultat à un multiple de 1, 2 ou 5, conformément aux spécifications du paramètre "cadence". Le poids total est stocké dans le registre à virgule flottante correspondant au numéro de la balance.

Poids stocké = arrondi (poids, nombre de décimales, cadence)

FUNC

FUNC XXXX YYYY

MNÉMONIQUE : **FUNC**
OPÉRANDE XXXX : **Constante 7**
OPÉRANDE YYYY : **Registre entier contenant le numéro de la balance à régler**
CODE INSTRUCTION : **90**

DESCRIPTION :

Réglage automatique du zéro (de la tare).
L'opérande XXXX est la constante 7 : appel de la fonction de réglage automatique du zéro.
L'opérande YYYY est l'adresse du registre entier contenant le numéro de la balance.
La première balance est la balance 0, qui correspond à la première entrée analogique détectée.
Affichage d'un message : non.
Ne fait pas intervenir les piles (0).

Remarque : La balance correspondante doit être activée à l'aide de la FUNC 9. Dans le cas contraire, le réglage automatique du zéro ne sera pas exécuté.

☑ Exemple :

```
MOVCI   2           ;Charge la constante 2.
STOI    600         ;Stocke la constante dans le registre entier.
INK     touc        ;Détection d'une touche.
JZ      sous1       ;Si la touche a été activée, exécution de
                    ;FUNC. Sinon, saut à "sous1".
FUNC    7   600     ;Réglage automatique du zéro de la
                    ;balance dont le numéro figure dans le
                    :registre entier 600 (balance numéro 3 (la
                    ;première est la 0)).
sous1 .....
```

FUNC

FUNC XXXX YYYY

MNÉMONIQUE : **FUNC**
OPÉRANDE XXXX : **Constante 8**
OPÉRANDE YYYY : **Constante 0**
CODE INSTRUCTION : **90**

DESCRIPTION :

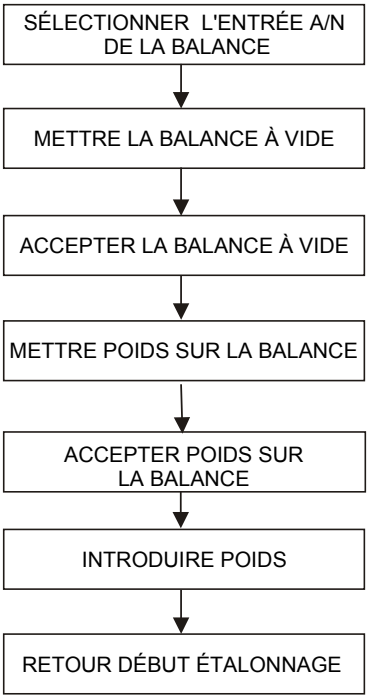
Étalonnage automatique de la balance.
L'opérande XXXX est la constante 8 : appel de la fonction d'étalonnage.
L'opérande YYYY est la constante 0.
Affichage d'un message : oui
Ne fait pas intervenir les piles (0).

Pour sélectionner l'entrée A/N de la balance à étalonner, utilisez les touches <↑> et <↓> du clavier de l'équipement MIDA.

Si la balance sélectionnée n'est pas activée, il est impossible de l'étalonner et un message (OFF) s'affiche en continu à l'écran. Si l'entrée A/N de la balance sélectionnée n'existe pas ou ne fonctionne pas, il est impossible de l'étalonner et le même message s'affiche à l'écran, mais en clignotant.

Remarque : La balance correspondante doit être activée à l'aide de la FUNC 9.

Le schéma du menu d'étalonnage de la balance est le suivant :


SÉLECTIONNER L'ENTRÉE A/N DE LA BALANCE
METTRE LA BALANCE À VIDE
ACCEPTER LA BALANCE À VIDE
METTRE POIDS SUR LA BALANCE
ACCEPTER POIDS SUR LA BALANCE
INTRODUIRE POIDS
RETOUR DÉBUT ÉTALONNAGE

FUNC

FUNC XXXX YYYY

MNÉMONIQUE : **FUNC**
OPÉRANDE XXXX : **Constante 9**
OPÉRANDE YYYY : **Mode de configuration de la balance**
CODE INSTRUCTION : **90**

DESCRIPTION :

Permet d'entrer dans le mode de configuration indiqué par l'opérande YYYY pour régler les paramètres de fonctionnement de la balance.
L'opérande XXXX est la constante 9 : appel de la fonction de configuration de la balance.
L'opérande YYYY est la constante qui indique quel mode de configuration vous souhaitez afficher :

0 Configuration utilisateur
1 Configuration avancée

Affichage d'un message : oui.
Ne fait pas intervenir les piles (0).

Pour sélectionner l'entrée A/N de la balance à étalonner, utilisez les touches <↑> et <↓> du clavier de l'équipement MIDA.

Si la balance sélectionnée n'est pas activée, il est impossible de l'étalonner et un message (OFF) s'affiche en continu à l'écran. Si l'entrée A/N de la balance sélectionnée n'existe pas ou ne fonctionne pas, il est impossible de l'étalonner et le même message s'affiche à l'écran, mais en clignotant.

Cette instruction permet d'activer les routines de pesage de n'importe quelle balance et de définir les paramètres de fonctionnement de celle-ci. Toutes ces données sont stockées dans l'EEPROM non volatile pour que les paramètres ne soient pas perdus lorsque vous éteignez l'équipement MIDA.

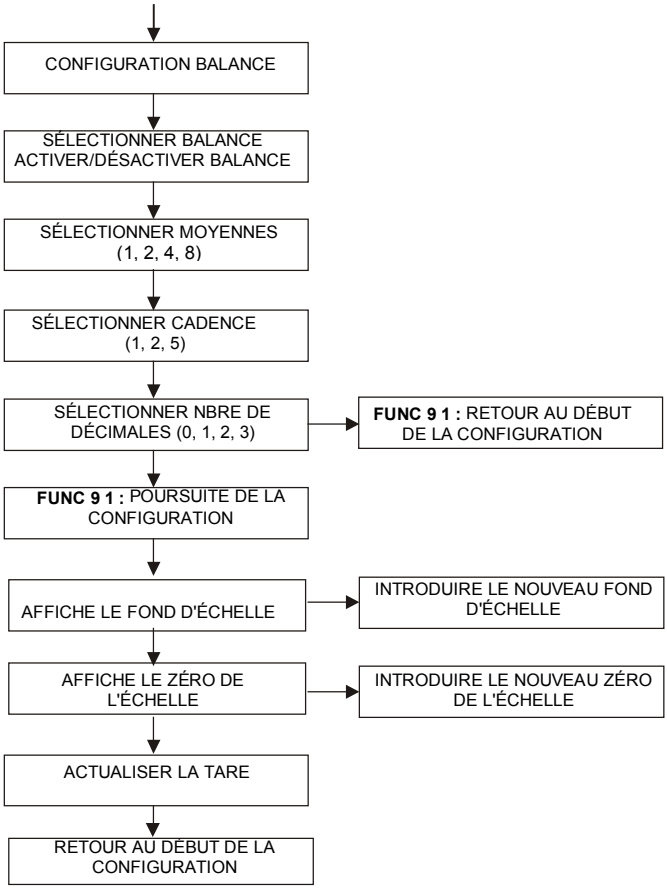
Pour configurer une balance, vous pouvez exécuter l'instruction suivante :

```
FUNC 9 1          ;Configuration avancée.
```

Pour ce faire, il vous faut connaître le facteur d'échelle et le zéro. L'obtention de ces paramètres peut également être automatisée grâce à la fonction (. L'exécution des instructions suivantes vous permet également de configurer une balance :

```
FUNC 9 0        ;Configuration utilisateur.
FUNC 8 0        ;Étalonnage automatique de la balance.
```

Le schéma de configuration de la balance est le suivant :



☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

```
;Exemple de programme pour MIDA 64.
;Exemple d'utilisation des fonctions de pesage.

;               Touche      Action
;               --------    ---------
;               F5          Exécute "FUNC 7 bal"
;               F6          Exécute "FUNC 8 0"
;               F7          Exécute "FUNC 9 0"
;               F8          Exécute "FUNC 9 1"
;               HAUT        Affichage de la balance suivante.
;               BAS         Affichage de la balance précédente.
;Tant qu'aucune touche n'est activée, afficher le numéro de balance et
;le poids mesuré.

;Définition des touches à utiliser

f5          equ   360   ;Relais correspondant à la touche <F5>.
f6          equ   361   ;Relais correspondant à la touche <F6>.
f7          equ   362   ;Relais correspondant à la touche <F7>.
f8          equ   363   ;Relais correspondant à la touche <F8>.
haut        equ   342   ;Relais correspondant à la touche <UP>.
bas         equ   343   ;Relais correspondant à la touche <DOWN>.

;Définition des registres entiers

bal         equ   400   ;Registre contenant le numéro de la balance.
frm         equ   401   ;Registre contenant le format d'affichage DISFX.

;Définition des libellés

texte0      lite  "BALANCE :"
texte1      lite  "POIDS :"
```

```
        SETRI   bal   0     ;Stocke la constante 0 dans le registre
                            ;entier "bal" (nº de balance).
        SETRI   frm   93    ;Stocke la constante 93 dans le registre
                            ;entier "frm" (format d'affichage).
debut   INK     f5          ;Détection de la touche <F5>.
        JZ      saut1
        FUNC    7     bal   ;Mise à zéro automatique de la balance
                            ;dont le nº est stocké dans le registre entier
                            ;"bal".
saut1   INK     f6          ;Détection de la touche <F6>.
        JZ      saut2
        FUNC    8     0     ;Étalonnage de la balance.
saut2   INK     f7          ;Détection de la touche <F7>.
        JZ      saut3
        FUNC    9     0     ;Paramètres utilisateur de la balance.
saut3   INK     f8          ;Détection de la touche <F8>.
        JZ      saut4
        FUNC    9     1     ;Paramètres avancés de la balance.
saut4   INK     haut        ;Détection de la touche <UP> (HAUT).
        JZ      saut5
        INC     bal   1     ;Incrémente de 1 unité le contenu du
                            ;registre entier "bal".
        MOVRI   bal         ;Charge le contenu du registre entier
                            ;"bal" dans la pile arithmétique.
        MOVCI   18          ;Charge la constante 18 dans la pile
                            ;arithmétique.
        CPLI    saut5       ;Si le contenu du registre entier "bal" n'est
                            ;pas 18, le programme saute à la ligne
                            ;"saut5".
        SETRI   bal   0     ;Si "bal" > = 18, "bal" = 0;
saut5   INK     bas         ;Détection de la touche <DOWN> (BAS).
        JZ      saut6
        INC     bal   -1    ;Décrémente de 1 unité le contenu du
                            ;registre entier "bal".
        MOVRI   bal         ;Charge le contenu du registre entier
                            ;"bal" dans la pile arithmétique.
        MOVCI   0           ;Charge la constante 0 dans la pile
                            ;arithmétique.
```

```
        CPGEI   saut6       ;Si le contenu du registre entier "bal" est
                            ;supérieur ou égal à 0, le programme saute
                            ;à la ligne "saut6".
        SETRI   bal    17   ;Si "bal" < 0, "bal" =17.
saut6   CLEAR               ;Efface la mémoire-tampon intermédiaire et
                            ;place le pointeur sur la première position.
        DISL    texte0      ;Charge le message de la table des libellés
                            ;"texte0" dans la mémoire-tampon
                            ;intermédiaire.
        LOC     14          ;Place le pointeur d'affichage sur la
                            ;quatorzième position (la première est la
                            ;position 0).
        DISRI   bal    2    ;Copie le contenu du registre entier "bal"
                            ;(à 2 chiffres), dans la mémoire-tampon
                            ;intermédiaire.
        LOC     16          ;Place le pointeur d'affichage sur la
                            ;seizième position (la première est la
                            ;position 0).
        DISL    texte1      ;Charge le message de la table des libellés
                            ;"texte1" dans la mémoire-tampon
                            ;intermédiaire.
        LOC     23          ;Place le pointeur d'affichage sur la
                            ;vingt-troisième position (la première est la
                            ;position 0).
        DISFX   bal    frm  ;Copie le contenu du registre à virgule
                            ;flottante signalé par le registre entier "bal"
                            ;dans la mémoire-tampon intermédiaire, au
                            ;format contenu dans le registre entier
                            ;"frm".
        COM     0           ;Copie le contenu de la mémoire-tampon
                            ;intermédiaire sur le LCD.
        JMP     debut       ;Saut inconditionnel à la ligne "début".
```

Certains équipements MIDA possèdent une fonction interne pour la programmation d'un contrôle PID. L'un de ses utilitaires est réservé au champ des applications qui requièrent l'intervention de contrôles de type Proportionnel Intégral Dérivé également connus sous le nom de fonctions de contrôle PID.

✓ Contrôle PID

Un contrôle PID est un système employé pour les processus de contrôle à rétroaction dans un système dit "en boucle fermée". La principale fonction du contrôleur PID est de faire en sorte que la sortie recherche et assure le suivi des variations du signal d'entrée en temps réel. La méthode permettant d'effectuer ces réglages est issue d'un système basé sur les contrôles de type Proportionnel Intégral Dérivé (PID).

La fonction PID intégrée à nos équipements permet d'éviter que le programmeur ait à effectuer tous ces calculs au cours d'un processus de régulation qui, comme vous le savez, est un processus au cours duquel une grandeur d'entrée donnée d'un système, variable dans le temps (grandeur réglée, valeur actuelle, etc.) et détectée en continu, est comparée avec une autre grandeur donnée (valeur prescrite, fixée, etc.) et influencée dans le sens d'une égalisation. Comme nous l'avons déjà commenté ci-dessus, le processus issu de ces opérations est exécuté dans un système en boucle fermée.

Les avantages de la fonction PID intégrée dans nos équipements MIDA sont évidents : toutes les opérations nécessaires à l'obtention d'un système de contrôle et de régulation de processus et à la garantie de sa stabilité deviennent inutiles grâce à ces fonctions de contrôle PID, ce qui représente un gain de temps considérable. Vous trouverez, ci-après, une description de ces fonctions.

✓ Programmation PID

La programmation du contrôle PID s'effectue par l'intermédiaire de l'instruction d'automate **FUNC** et de la manière suivante :

- La première opération consiste à configurer la fonction PID grâce à l'instruction **FUNC 1 n**, où n est le numéro de PID à utiliser (compris entre 0 et 9 - nombre maximal de contrôles PID : 10).

- N'oubliez pas qu'il est nécessaire de remettre les variables internes de la fonction PID à zéro grâce à l'instruction **FUNC 2 n** (où n est le numéro de PID à utiliser) avant d'effectuer le contrôle PID.

- L'opération suivante consiste à effectuer le contrôle PID proprement dit grâce à l'instruction **FUNC 0 n**, où n est le numéro de PID à utiliser.

Arrivés là, vous aurez remarqué qu'il existe trois instructions - FUNC 0, FUNC 1 et FUNC 2 - se référant au contrôle PID. Nous les détaillons ci-après.

FUNC

FUNC XXXX YYYY

MNÉMONIQUE : **FUNC**
OPÉRANDE XXXX : **Constante 0**
OPÉRANDE YYYY : **Numéro de PID à utiliser**
CODE INSTRUCTION : **90**

DESCRIPTION :

Effectue le contrôle PID.
L'opérande XXXX est la constante 0 : appel de la fonction de contrôle PID.
L'opérande YYYY indique le numéro de PID à utiliser (de 0 à 9).
Décharge de la pile arithmétique le contenu d'un registre entier préalablement chargé (valeur de la mesure réalisée) pour effectuer ultérieurement le contrôle PID et placer le résultat sur la pile arithmétique.
N'altère pas la pile arithmétique.

Remarque : L'erreur (Consigne – Mesure) et la valeur de contrôle sont normalisées : de -10000 à 10000.

☑ Exemple :

Reportez-vous à la fin du chapitre.

FUNC

FUNC XXXX YYYY

MNÉMONIQUE : **FUNC**
OPÉRANDE XXXX : **Constante 1**
OPÉRANDE YYYY : **Numéro de PID à configurer**
CODE INSTRUCTION : **90**

DESCRIPTION :

Configuration des paramètres du PID.
L'opérande XXXX est la constante 1: appel de la fonction de configuration des paramètres du PID.
L'opérande YYYY indique le numéro de PID à configurer (de 0 à 9).
Décharge de la pile arithmétique la valeur indiquant l'adresse de début des paramètres PID:

Adresse début (registre entier x):	valeur Consigne
Adresse x+1 (registre entier x+1):	valeur K*100
Adresse x+2 (registre entier x+2):	valeur P_I
Adresse x+3 (registre entier x+3):	valeur P_D*10

Les adresses (registrers entiers) antérieurs doivent être consécutives et figurer dans l'ordre indiqué. Les valeurs du PID, sont celles de ces registres entiers.
Décrémente la pile arithmétique de 1 niveau (-1).

Remarque : Toutes les valeurs sont normalisées : de -10000 à 10000.

Étant donné que la valeur stockée dans Adresse+1 est K*100, K varie de -100.00 à 100.00 ; le signe négatif indique les systèmes à action inverse (une erreur négative donne une sortie positive).

P_I peut être compris entre 0 et 10000 ; 0 indique l'absence d'action intégrale.

Étant donné que la valeur stockée dans Adresse+3 est P_D*10, P_D varie de 0 à 1000.0 (P_D est généralement bien inférieur à P_I). Une valeur de P_D égale à 0 indique l'absence d'action différentielle.

$P_I = T_R/T_o$ T_R - Durée du réajustement intégral.

$P_D = T_A/T_o$

T_A - Durée de l'avance dérivative.
T_o - Durée d'échantillonnage.

L'adresse inclut uniquement des registres RAM non volatile. L'emploi de registres non volatiles d'EEPROM entraîne l'apparition d'une erreur.

☑ Exemple :

Reportez-vous à la fin du chapitre.

FUNC

FUNC XXXX YYYY

MNÉMONIQUE : **FUNC**
OPÉRANDE XXXX : **Constante 2**
OPÉRANDE YYYY : **Numéro de PID à initialiser**
CODE INSTRUCTION : **90**

DESCRIPTION :

Initialisation des variables internes du contrôle PID.
L'opérande XXXX est la constante 2 : appel de la fonction d'initialisation du contrôle PID.
L'opérande YYYY indique le numéro de PID à initialiser.
Ne fait pas intervenir les piles (0).

☑ Exemple :

Reportez-vous à la fin du chapitre.

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A. Contrôle PID pour un système thermique.

Supposons que nous ayons, d'une part, une sonde destinée à mesurer la température à travers une entrée analogique (en points de convertisseur et non en degrés centigrades).

Et d'autre part, une résistance destinée à chauffer le système, mais pouvant uniquement être activée et désactivée.

Le contrôle PID nous fournit une valeur de contrôle. La sortie est contrôlée par largeur d'impulsion. La résistance est activée pendant toute la durée To qui s'écoule entre deux mesures ; le reste du temps, la résistance est désactivée.

```
;Définition des étiquettes

relais      equ   500       ;Relais des temporisateurs.
sortie      equ   100       ;Sortie numérique qui active la résistance.
;cns, K, Pi, Pd doivent être consécutif et figurer dans cet ordre.
cns         equ   300       ;Reg. entier qui contient la valeur de
                            ;consigne.
K           equ   301       ;Reg. entier qui contient la valeur de K
Pi          equ   302       ;Reg. entier qui contient la valeur de Pi
Pd          equ   303       ;Reg. entier qui contient la valeur de Pd
offs        equ   19843     ;Offset déduit de la mesure.
tmm         equ   400       ;Adresse de stockage temporaire de la
                            ;mesure.
controle    equ   401       ;Adresse de stockage de la variable de
                            ;contrôle.
tmp         equ   402       ;Adresse de stockage temporaire des
                            ;calculs.
timer       equ   250       ;Temporisateur.
mesure      equ   60        ;Entrée analogique.

;Dans cet exemple, le contrôle PID fonctionne en boucle fermée.
;Il lit l'entrée, exécute la fonction de contrôle PID et active la sortie en
;fonction de la réponse.
```

```
        MOVCI   cns         ;Charge la constante "cns" dans la pile
                            ;arithmétique.
        FUNC    1     0     ;Décharge de la pile arithmétique la valeur
                            ;indiquant l'adresse initiale des paramètres
                            ;du PID 0 (important).
        FUNC    2     0     ;Initialise les variables internes du PID 0.
bouc    MOVRI   mesure      ;Charge le registre entier avec la valeur de
                            ;l'entrée analogique dans la pile
                            ;arithmétique.
        SUBC    offs        ;Déduit la constante indiquée (équivalente
                            ;à l'offset calculé au cours d'une routine
                            ;d'étalonnage).
        STOI    tmm         ;Stocke le résultat dans le registre entier
                            ;"tmm".
        MOVRI   tmm         ;Charge le contenu du registre entier "tmm"
                            ;dans la pile arithmétique.
        FUNC    0     0     ;Décharge le contenu de la pile
                            ;arithmétique pour effectuer le contrôle du
                            ;PID 0 et place le résultat sur cette même
                            ;pile arithmétique.
        STOI    controle    ;Stocke le contenu de la pile arithmétique
                            ;dans le registre entier "contrôle".
        CALL    affich      ;Appelle la routine "affich".
        CALL    rout_ct     ;Appel la routine "rout_ct".
        JMP     bouc        ;Saut inconditionnel à "bouc".

;Affiche la variable d'entrée et la variable de contrôle à l'écran.

affich  CLEAR               ;Efface la mémoire-tampon intermédiaire et
                            ;place le pointeur sur la première position.
        DISRI   tmm   4     ;Charge le contenu du registre entier "tmm"
                            ;dans la mémoire-tampon intermédiaire,
                            ;dans le format spécifié par le deuxième
                            ;opérande.
        LOC     5           ;Positionne le pointeur d'affichage.
        MOVRI   controle    ;Charge le contenu du registre entier
                            ;"contrôle" dans la pile arithmétique.
        DIVC    100         ;Divise par la constante indiquée (passe en
                            ;pourcentage).
```

```
        STOI    tmp     ;Stocke le contenu de la pile arithmétique
                        ;dans le registre entier "tmp".
        DISRI   tmp  2  ;Charge le contenu du registre entier "tmp"
                        ;dans la mémoire-tampon intermédiaire,
                        ;dans le format spécifié par le deuxième
                        ;opérande..
        COM     0       ;Affiche le contenu de la mémoire-tampon
                        ;intermédiaire sur le LCD.
        RET             ;Retour de la routine.
```

;Contrôle de la sortie à partir de la largeur d'impulsion.
;Les temporisateurs du MIDA utilisent une base de temps de 100 ms.
;En divisant la variable de contrôle (0..10.000) par 100, nous obtenons
;un contrôle de temps de 0 à 10 secondes.

```
rout_ct MOVRI   controle ;Charge le contenu du
                         ;registre entier "contrôle" dans la
                         ;pile arithmétique.
        MOVCI   0        ;Charge la constante 0 dans la
                         ;pile arithmétique.
        CPLI    c_neg    ;Compare la constante 0 et le
                         ;contenu du registre "contrôle". Si
                         ;celui-ci est négatif, saute à "c-nég".
        MOVRI   controle ;Charge le contenu du registre
                         ;entier "contrôle" (obtention d'une
                         ;valeur comprise entre 0 et 10
                         ;secondes) dans la pile
                         ;arithmétique.
        DIVC    100      ;Divise par la constante 100.
        STOI    tmp      ;Charge le contenu du registre
                         ;entier "tmp" dans la pile
                         ;arithmétique.
        JMP     ctr      ;Saut inconditionnel à "ctr".
c_neg   MOVCI   0        ;Charge la constante 0 dans la pile
                         ;arithmétique.
        STOI    tmp      ;Stocke le contenu de la pile
                         ;arithmétique dans le registre entier
                         ;"tmp".
ctr     SET     sortie   ;Active le relais "sortie" (active la
                         ;sortie).
```

```
        SET     relais      ;Active le relais "relais"
tim1    LDNT    relais      ;Charge l'état complémenté du
                            ;relais "relais" dans la pile logique.
        TIMR    timer  tmp  ;Temporise tmp/10 secondes.
        OUT     relais      ;Décharge le résultat du
                            ;temporisateur sur le relais "relais".
        LD      relais      ;Charge l'état du relais "relais" dans
                            ;la pile logique.
        JZ      tim1        ;Saute si le "timer" n'a pas terminé.
        MOVCI   100         ;Charge la constante 100 dans la
                            ;pile arithmétique.
        MOVRI   tmp         ;Charge le contenu du registre
                            ;entier "tmp" dans la pile
                            ;arithmétique.
        SUBI                ;Soustrait les données contenues
                            ;dans la pile arithmétique.
        STOI    tmp         ;Stocke le résultat de la
                            ;soustraction dans le registre entier
                            ;"tmp" (100 - tmp).
        RESET   sortie      ;Désactive le relais "sortie"
                            ;(désactive la sortie).
        SET     relais      ;Active le relais "relais".
tim2    LDNT    relais      ;Charge l'état complémenté du
                            ;relais "relais" dans la pile logique.
        TIMR    timer  tmp  ;Temporise le reste jusqu'à 10
                            ;secondes.
        OUT     relais      ;Décharge le résultat du
                            ;temporisateur sur le relais "relais".
        LD      relais      ;Charge l'état du relais "relais" dans
                            ;la pile logique.
        JZ      tim2        ;Saute si le "timer" n'a pas terminé.
        RET                 ;Retour de la routine.
```

DF

✓ POINTEURS ET TABLES DE REGISTRES (ARRAYS)

Les équipements MIDA disposent d'un jeu d'instructions dites "d'adressage indirect".
Ce jeu d'instructions permet de créer ou d'utiliser des pointeurs pour indexer les tables de registres (arrays) ou de messages.

✓ INSTRUCTIONS D'ADRESSAGE INDIRECT

Le tableau suivant contient les instructions d'adressage indirect de l'équipement :

LDX	**MOVIX**	**DISIX**	**INPIX**
OUTX	**STOIX**	**DISFX**	**INFPX**
	MOVFX	**DISLX**	**INPCX**
	STOFX	**LOCX**	

Bien que nous ayons déjà décrit chacune de ces instructions dans le chapitre "IN" de ce manuel, nous reviendrons, dans le présent chapitre, sur certains aspects concernant l'emploi des instructions indexées.

✓ OPÉRATIONS LIÉES À CES INSTRUCTIONS

Les opérations pouvant être réalisées grâce aux instructions ci-dessus sont les suivantes :

- ***Manipulation d'une table de relais (chargement et déchargement).***
- ***Manipulation d'une table de registres entiers ou à virgule flottante (chargement et déchargement).***
- ***Affichage de messages et de registres entiers ou à virgule flottante par adressage indirect.***

- ***Positionnement du curseur d'affichage par adressage indirect.***
- ***Introduction de données numériques entières ou à virgule flottante par adressage indirect.***

✓ **Mode de programmation**

Nous avons déjà décrit la manière d'utiliser ce groupe d'instructions dans le paragraphe correspondant de ce manuel.

Les éléments logiciels employés sont des pointeurs et des tables de registres et de messages.

✓ **Manipulation d'une table de relais (LDX, OUTX).**

Les instructions LDX et OUTX chargent et déchargent l'état des relais de la pile logique.

L'adresse du relais chargé ou déchargé est celle qui figure dans le registre entier qui fait office de pointeur.

Le déchargement de l'état de l'entrée 0 sur la sortie 101 peut être effectué de la manière suivante :

```
SETRI    pointeur1   0     ;Stocke la donnée 0 dans le registre
                           ;entier "pointeur1".
SETRI    pointeur2   101   ;Stocke la donnée 101 dans le
                           ;registre entier "pointeur2".
LDX      pointeur1         ;Charge, dans la pile logique, l'état
                           ;du relais indiqué dans le registre
                           ;entier "pointeur1" (dans ce cas,
                           ;l'entrée numérique 0).
OUTX     pointeur2         ;Décharge de la pile logique l'état
                           ;de l'entrée 0 et le charge sur le
                           ;relais désigné par le registre entier
                           ;"pointeur2" (dans ce cas, la sortie
                           ;numérique 101).
```

Les deux premières lignes de ce programme permettent d'attribuer aux registres entiers la valeur des adresses des relais qu'ils "*désignent*".

✓ **Manipulation des tables de registres (MOVIX, STOIX, MOVFX, STOFX*).***

Ces instructions permettent de charger et de décharger des données entières ou à virgule flottante de la pile arithmétique.

L'adresse du registre chargé ou déchargé est celle qui figure dans le registre entier qui fait office de pointeur.

Le transfert de données d'un registre à un autre peut être effectué de la manière suivante :

```
SETRI    pointeur1   500   ;Stocke la donnée 500 dans le
                           ;registre entier "pointeur1".
SETRI    pointeur2   600   ;Stocke la donnée 600 dans le
                           ;registre entier "pointeur2".
SETRI    pointeur3   0     ;Stocke la donnée 0 dans le
                           ;registre entier "pointeur3".
SETRI    pointeur4   100   ;Stocke la donnée 100 dans le
                           ;registre entier "pointeur4".
```

;Transfert entre registres entiers.

```
MOVIX    pointeur1   ;Charge, dans la pile arithmétique, le
                     ;contenu du registre entier désigné par le
                     ;registre "pointeur1", c'est-à-dire le contenu
                     ;du registre entier 500.
STOIX    pointeur2   ;Stocke le contenu de la pile arithmétique
                     ;dans le registre entier désigné par le
                     ;registre "pointeur2", c'est-à-dire dans le
                     ;registre entier 600.
```

;Transfert entre registres à virgule flottante.

```
MOVFX    pointeur3   ;Charge, dans la pile arithmétique, le
                     ;contenu du registre à virgule flottante
                     ;désigné par le registre entier "pointeur3",
                     ;c'est-à-dire le contenu du registre à virgule
                     ;flottante 0.
```

```
STOFX    pointeur4   ;Stocke le contenu de la pile arithmétique
                     ;dans le registre à virgule flottante désigné
                     ;par le registre entier "pointeur4", c'est-à-
                     ;dire dans le registre à virgule flottante 100.
```

Les quatre premières lignes de ce programme permettent d'attribuer aux registres entiers la valeur des adresses des registres qu'ils "*désignent*".

✓ Positionnement du curseur et affichage de registres et de messages (LOCX, DISLX, DISIX, DISFX)

Toutes les opérations d'affichage (et de transmission) peuvent être effectuées par adressage indirect :

- Le curseur d'affichage peut être placé sur une position de la mémoire-tampon intermédiaire désignée par le contenu d'un registre.
- Il est possible d'afficher le contenu de registres dont l'adresse figure dans un autre registre pointeur.
- Il est également possible d'afficher des textes dont l'adresse figure dans un registre.

Vous trouverez, ci-après, un programme contenant ces trois opérations :

```
;Initialisation des pointeurs.

SETRI    pointeur1   10    ;Stocke la donnée 10 dans le
                           ;registre entier "pointeur1".
SETRI    pointeur2   400   ;Stocke la donnée 400 dans le
                           ;registre entier "pointeur2".
SETRI    pointeur3   20    ;Stocke la donnée 20 dans le
                           ;registre entier "pointeur3".

;Affichage et transmission via un port de communication..
```

```
CLEAR                                ;Efface la mémoire-tampon
                                     ;intermédiaire et place le pointeur
                                     ;sur la première position.
LOCX    pointeur1                    ;Place le pointeur d'affichage sur la
                                     ;position indiquée par le contenu du
                                     ;registre entier "pointeur1", c'est-à-
                                     ;dire sur la position 10.
DISIX   pointeur2   form             ;Copie, dans la mémoire-tampon
                                     ;intermédiaire, le contenu du
                                     ;registre entier désigné par le
                                     ;registre entier "pointeur2", dans le
                                     ;format indiqué par le contenu du
                                     ;registre entier "form".
DISLX   pointeur3                    ;Copie, dans la mémoire-tampon
                                     ;intermédiaire, le texte indiqué par
                                     ;le contenu du registre entier
                                     ;"pointeur3".
DISFX   pointeur2   form             ;Copie, dans la mémoire-tampon
                                     ;intermédiaire, le contenu du
                                     ;registre à virgule flottante désigné
                                     ;par le registre entier "pointeur2",
                                     ;dans le format indiqué par le
                                     ;contenu du registre entier "form".
COM     0                            ;Copie le contenu de la mémoire-
                                     ;tampon intermédiaire sur le LCD.
COM     1                            ;Transmet le contenu de la
                                     ;mémoire-tampon intermédiaire via
                                     ;le port COM1.
```

✓ ***Introduction de données numériques entières et à virgule flottante*** **(INPX, INPFX, INPCX)**

L'adresse du registre contenant la donnée est celle qui figure dans le registre entier qui fait office de pointeur.

Le même registre pointeur peut être utilisé pour les instructions décrites précédemment.

Le format d'introduction des données est compatible avec le format des instructions DISIX et DISFX.

EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Chargement de la donnée 3 dans une table de registres entiers (de 400 à 490).

```
;Définition des libellés
texte0      lite    “EXEMPLE-A DF-AI”
texte1      lite    “EXEMPLE RUN”

        LD          340             ;Détecte si la touche correspondante est
                                    ;activée.
        JZ          fin             ;Dans le cas contraire, le programme saute
                                    ;à "fin".
        SETRI       300     490     ;Stocke la donnée 490 dans le registre
                                    ;entier 300.
        SETRI       301     400     ;Stocke la donnée 400 dans le registre
                                    ;entier 301.
        SETRI       302     3       ;Stocke la donnée 3 dans le registre entier
                                    ;3.
eti1    MOVRI       302             ;Charge le contenu du registre entier 302
                                    ;dans la pile arithmétique.
        STOIX       301             ;Stocke le contenu de la pile arithmétique
                                    ;dans le registre signalé par le registre
                                    ;entier 301, c'est-à-dire dans le registre
                                    ;entier 400.
        INC         301     1       ;Incrémente de 1 unité le contenu du
                                    ;registre entier 301 (incrémente le pointeur
                                    ;dans la table des registres).
        MOVRI       301             ;Charge le contenu du registre entier 301
                                    ;dans la pile arithmétique.
        MOVRI       300             ;Charge le contenu du registre entier 300
                                    ;dans la pile arithmétique.
        CPLEI       eti1            ;Si le contenu du registre entier 301 est
                                    ;inférieur ou égal au contenu du registre
                                    ;entier 300, le programme saute à "éti1".
                                    ;Dans le cas contraire, son exécution se
                                    ;poursuit (il détecte la fin du chargement
                                    ;de la table).
```

```
        CLEAR                   ;Efface la mémoire-tampon intermédiaire et
                                ;place le pointeur sur la première position.
        DISL    texte1          ;Copie, dans la mémoire-tampon
                                ;intermédiaire, le message de la table de
                                ;libellés "texte1".
        COM     0               ;Copie le contenu de la mémoire-tampon
                                ;intermédiaire sur le LCD.
        JMP     fin1            ;Saut inconditionnel à "fin1".
fin     CLEAR                   ;Efface la mémoire-tampon intermédiaire et
                                ;place le pointeur sur la première position.
        DISL    texte0          ;Copie, dans la mémoire-tampon
                                ;intermédiaire, le message de la table de
                                ;libellés "texte0".
        COM     0               ;Copie le contenu de la mémoire-tampon
                                ;intermédiaire sur le LCD.
fin1    END                     ;Fin du programme.
```

B.- Affichage du mois en cours sur le LCD de l'équipement.

```
;Définition des libellés
texte0      lite    " "
texte1      lite    "JANVIER"
texte2      lite    "FÉVRIER"
texte3      lite    "MARS"
texte4      lite    "AVRIL"
texte5      lite    "MAI"
texte6      lite    "JUIN"
texte7      lite    "JUILLET"
texte8      lite    "AOÛT"
texte9      lite    "SEPTEMBRE"
texte10     lite    "OCTOBRE"
texte11     lite    "NOVEMBRE"
texte12     lite    "DÉCEMBRE"
;
        CLEAR           ;Efface la mémoire-tampon intermédiaire et
                        ;place le pointeur sur la première position.
```

```
        DISLX     46    ;Copie, dans la mémoire-tampon intermédiaire, le
                        ;texte dont le numéro d'ordre figure dans le
                        ;registre entier 46 (adresse du registre entier
                        ;correspondant au "Mois de l'année").
        COM       0     ;Copie le contenu de la mémoire-tampon
                        ;intermédiaire sur le LCD.
        END             ;Fin du programme.
```

Les messages sont associés à une adresse qui coïncide avec l'ordre dans lequel ils ont été déclarés. Dans l'exemple ci-dessus, nous avons choisi d'assigner à chaque texte une étiquette indiquant le numéro d'ordre de chacun d'entre eux, mais il aurait pu s'agir de toute étiquette valide.

C.- Détection du relais activé (de 400 à 500), en supposant qu'un seul d'entre eux l'est. Affichage de l'adresse du relais détecté. Si aucun relais activé n'est détecté, affichage d'un message à l'écran.

```
;Définition des étiquettes.
point1      equ   600          ;Registre entier pour le pointeur 1.

;Définition des textes.
detect      lite  “RELAIS ACTIVÉ”
nodet1      lite  “RELAIS”
nodet2      lite  “NON TROUVÉ”

        MOVCI     400          ;Charge la constante 400 dans la pile
                               ;arithmétique.
        STOI      point1       ;Stocke le contenu de la pile arithmétique
                               ;dans le registre entier "point1" (avec
                               ;l'adresse du premier relais interne de la
                               ;table des relais).
        SET       600          ;Active le relais interne 600 (indique qu'un
                               ;relais activé a été découvert).
eti1    LDX       point1       ;Charge, dans la pile logique, l'état du
                               ;relais indiqué dans le registre entier
                               ;"point1".
```

```
        JNZ     eti2        ;Détecte si le relais signalé par "point1"
                            ;est activé.
        INC     point1 1    ;Incrémente de 1 unité le contenu du
                            ;registre entier "point1" (incrémente le
                            ;pointeur de la table des relais).
        MOVRI   point1      ;Charge le contenu du registre entier
                            ;"point1" dans la pile arithmétique.
        MOVCI   500         ;Charge la constante 500 dans la pile
                            ;arithmétique.
        CPLEI   eti1        ;Détecte si le transfert de données est
                            ;terminé. Vérifie si la constante 500 est
                            ;inférieure ou égale au registre entier
                            ;"point1". Le cas échéant, le programme
                            ;saute à "éti1".
        RESET   600         ;Désactive le relais 600 si aucun relais actif
                            ;n'a été détecté dans la table.

;Affichage du résultat de l'exploration de la table des relais.

eti2    CLEAR               ;Efface la mémoire-tampon
                            ;intermédiaire et place le pointeur sur la
                            ;première position.
        LD      600         ;Charge l'état du relais 600 dans la pile
                            ;logique.
        JZ      eti3        ;Détecte si un relais activé a été découvert.
                            ;Dans le cas contraire, le programme saute
                            ;à "éti3".

;Affichage du relais découvert et de son adresse.

        CLEAR               ;Efface la mémoire-tampon intermédiaire et
                            ;place le pointeur sur la première position.
        DISL    detect      ;Copie, dans la mémoire-tampon
                            ;intermédiaire, le message de la table de
                            ;libellés "détect".
        LOC     21          ;Place le pointeur dans la position
                            ;indiquée.
```

```
        DISRI   point1 3    ;Copie le contenu du registre entier ;
                            "point1" dans la mémoire-tampon
                            ;intermédiaire, dans le format indiqué
                            ;par le deuxième opérande.
        JMP     eti4        ;Saut inconditionnel à "éti4".
```

;Affichage du relais non découvert.

```
eti3    LOC     5           ;Place le pointeur dans la position
                            ;indiquée.
        DISL    nodet1      ;Copie, dans la mémoire-tampon
                            ;intermédiaire, le message de la table de
                            ;libellés "nodét1".
        LOC     17          ;Place le pointeur dans la position
                            ;indiquée.
        DISL    nodet2      ;Copie, dans la mémoire-tampon
                            ;intermédiaire, le message de la table de
                            ;libellés "nodét2".
eti4    COM     0           ;Copie le contenu de la mémoire-tampon
                            ;intermédiaire sur le LCD.
        END                 ;Fin du programme.
```

D.- Transfert de données entre deux registres entiers.
La table A va du registre 400 au registre 450 et la table B du registre 500 au registre 550.

;Définition des étiquettes.

```
point1      equ     600     ;Registre entier pour le pointeur 1.
point2      equ     602     ;Registre entier pour le pointeur 2.
```

;Définition des libellés.

```
texte0      lite    “EXEMPLE-D DF-AI”
texte1      lite    “EXEMPLE RUN”
```

```
        LD      340         ;Détecte si la touche correspondante est
                            ;activée.
        JZ      fin         ;Dans le cas contraire, le programme saute
                            ;à "fin".
```

```
        MOVCI   400     ;Charge la constante 400 dans la pile
                        ;arithmétique.
        STOI    point1  ;Stocke le contenu de la pile arithmétique
                        ;dans le registre entier "point1" (adresse du
                        ;premier registre de la table A).
        MOVCI   500     ;Charge la constante 500 dans la pile
                        ;arithmétique.
        STOI    point2  ;Stocke le contenu de la pile arithmétique
                        ;dans le registre entier "point2" (adresse du
                        ;premier registre de la table B).
eti1    MOVIX   point1  ;Charge, dans la pile arithmétique, le
                        ;contenu du registre entier signalé par le
                        ;registre entier "point1".
        STOIX   point2  ;Stocke le contenu de la pile arithmétique
                        ;dans le registre entier signalé par le
                        ;registre entier "point2".
        INC     point1 1;Incrémente de 1 unité le registre entier
                        ;"point1" (incrémente le pointeur de la table
                        ;A).
        INC     point2 1;Incrémente de 1 unité le contenu du registre
                        ;entier "point2" (incrémente le pointeur de la
                        ;table B).
        MOVRI   point1  ;Charge le contenu du registre entier
                        ;"point1" dans la pile arithmétique.
        MOVCI   450     ;Charge la constante 450 dans la pile
                        ;arithmétique.
        CPLEI   eti1    ;Détecte si le transfert de données est
                        ;terminé. Vérifie si la constante 450 est
                        ;inférieure ou égale au registre entier
                        ;"point1". Le cas échéant, le programme
                        ;saute à "éti1".
        CLEAR           ;Efface la mémoire-tampon intermédiaire et
                        ;place le pointeur sur la première position.
        DISL    texte1  ;Copie, dans la mémoire-tampon
                        ;intermédiaire, le message de la table de
                        ;libellés "texte1".
        COM     0       ;Copie le contenu de la mémoire-tampon
                        ;intermédiaire sur le LCD.
        JMP     fin1    ;Saut inconditionnel à "fin1".
```

```
fin     CLEAR                ;Efface la mémoire-tampon intermédiaire et
                             ;place le pointeur sur la première position.
        DISL        texte0   ;Copie, dans la mémoire-tampon
                             ;intermédiaire, le message de la table de
                             ;libellés "texte0".
        COM         0        ;Copie le contenu de la mémoire-tampon
                             ;intermédiaire sur le LCD.
fin1    END                  ;Fin du programme.
```

E.- Transfert de données d'une table de registres à virgule flottante à une autre table de registres entiers.
La table A (à virgule flottante) va du registre 100 au registre 110 et la table B (format entier) du registre 400 au registre 410.
Le transfert doit avoir lieu de registre à registre grâce à l'activation d'une touche et le contenu des registres d'origine et des registres de destination doit s'afficher.

```
;Définition des étiquettes.
point1      equ   600      ;Registre entier pour le pointeur 1.
point2      equ   602      ;Registre entier pour le pointeur 2.
inter       equ   603      ;Registre entier pour les données
                           ;intermédiaires.
form        equ   605      ;Registre entier pour le format.

;Initialisation des registres de pointeurs.

        SETRI     form  51 ;Stocke la donnée 51 dans le registre entier
                           ;"form".
        MOVCI     100      ;Charge la constante 100 dans la pile
                           ;arithmétique.
        STOI      point1   ;Stocke le contenu de la pile arithmétique
                           ;dans le registre entier "point1" (adresse du
                           ;premier registre de la table A).
        MOVCI     400      ;Charge la constante 400 dans la pile
                           ;arithmétique.
        STOI      point2   ;Stocke le contenu de la pile arithmétique
                           ;dans le registre entier "point2" (adresse du
                           ;premier registre de la table B).
```

;Boucle de transfert de données.

```
eti1    INK       340          ;Détection d'une touche.
        JZ        eti1         ;Attend que la touche soit activée pour
                               ;effectuer le transfert de chaque donnée.
                               ;Si la touche n'est pas activée, le
                               ;programme saute à "éti1".
        MOVFX     point1       ;Charge, dans la pile arithmétique, le
                               ;contenu du registre à virgule flottante
                               ;signalé par le registre entier "point1".
        STOFI     inter        ;Stocke le contenu de la pile arithmétique
                               ;dans le registre entier "inter", l'arrondit et
                               ;le convertit au format entier.
        MOVRI     inter        ;Charge le contenu du registre entier "inter"
                               ;dans la pile arithmétique.
        STOIX     point2       ;Stocke le contenu de la pile arithmétique
                               ;dans le registre entier signalé par le
                               ;registre "point2".
```

;Affichage du programme.

```
        CLEAR                  ;Efface la mémoire-tampon intermédiaire et
                               ;place le pointeur sur la première position.
        DISFX     point1 form  ;Copie le contenu du registre à virgule
                               ;flottante signalé par le registre entier
                               ;"point1" dans la mémoire-tampon
                               ;intermédiaire, dans le format indiqué par
                               ;le contenu du registre entier "form".
        LOC       16           ;Place le pointeur dans la position
                               ;indiquée.
        DISIX     point2 form  ;Copie le contenu du registre entier signalé
                               ;par le registre entier "point2" dans la
                               ;mémoire-tampon intermédiaire, dans le
                               ;format indiqué par le contenu du registre
                               ;entier "form".
        COM       0            ;Copie le contenu de la mémoire-tampon
                               ;intermédiaire sur le LCD.
```

;Contrôle des pointeurs dans les deux tables.

```
        INC     point1 1    ;Incrémente de 1 unité le contenu du
                            ;registre entier "point1" (incrémente le
                            ;pointeur de la table A).
        INC     point2 1    Incrémente de 1 unité le contenu du
                            ;registre entier "point2" (incrémente le
                            ;pointeur de la table B).
        MOVRI   point1      ;Charge le contenu du registre entier
                            ;"point1" dans la pile arithmétique.
        MOVCI   110         ;Charge la constante 110 dans la pile
                            ;arithmétique.
        CPLEI   eti1        ;Détecte si le transfert de données est
                            ;terminé. Vérifie si la constante 110 est
                            ;inférieure ou égale au registre entier
                            ;"point1". Le cas échéant, le programme
                            ;saute à "éti1".
```

F.- Introduction de cinq données entières dans les registres entiers 400 à 404, sans interrompre l'exécution du programme et avec les textes correspondants.

```
;Définition des étiquettes.
expl        equ     399     ;Relais interne "Relais de 1ère exploration".
r_in        equ     391     ;Relais interne "Relais entrée".
point1      equ     600     ;Registre entier pour le pointeur 1.
point2      equ     602     ;Registre entier pour le pointeur 2.
form        equ     605     ;Registre entier pour le format.
;Définition des libellés.
texte0      lite    "Consigne 1"
texte1      lite    "Consigne 2"
texte2      lite    "Consigne 3"
texte3      lite    "Consigne 4"
texte4      lite    "Consigne 5"
texte5      lite    "Traitement en cours"

        LD      expl        ;Charge l'état du relais "expl " dans la pile
                            ;logique.
```

```
        JNZ     ini         ;Initialise les pointeurs pendant la première
                            ;exploration du programme. Saute à "ini"
                            ;lors de l'exploration suivante.
        SETRI   point1 400  ;Stocke la donnée 400 dans le registre
                            ;entier "point1".
        SETRI   point2 0    ;Stocke la donnée 0 dans le registre
                            ;entier "point2".
        SETRI   form   4    ;Stocke la donnée 4 dans le registre
                            ;entier "form".
ini     INK     356         ;Détection d'une touche.
        JZ      fin         ;Attend que la touche soit activée pour
                            ;effectuer l'introduction de chaque donnée.
                            ;Si la touche n'est pas activée, le
                            ;programme saute à "fin".
        LD      r_in        ;Charge l'état du relais "r_in" dans la pile
                            ;logique.
        JNZ     fin         ;Si le relais "r_in" est activé (Relais entrée),
                            ;le programme saute à "fin". Dans le cas
                            ;contraire, son exécution se poursuit par
                            ;l'introduction des données suivantes.
        CLEAR               ;Efface la mémoire-tampon intermédiaire et
                            ;place le pointeur sur la première position.
        DISLX   point2      ;Copie, dans la mémoire-tampon
                            ;intermédiaire, le texte dont le numéro
                            ;d'ordre figure dans le registre entier
                            ;"point2" (pointeur de textes).
        COM     0           ;Copie le contenu de la mémoire-tampon
                            ;intermédiaire sur le LCD.
        INPIX   point1 form ;Introduction d'une donnée dans le registre
                            ;entier signalé par le registre "point1"
                            ;(pointeur de registres), dans le format
                            ;indiqué dans le registre entier "form".
        INC     point1 1    ;Incrémente de 1 unité le contenu du
                            ;registre entier "point1" (incrémente le
                            ;pointeur de la table des registres).
        INC     point2 1    ;Incrémente de 1 unité le contenu du
                            ;registre entier "point2" (incrémente le
                            ;pointeur de la table des textes).
        MOVRI   point2      ;Charge le contenu du registre entier
                            ;"point2" dans la pile arithmétique.
```

```
        MOVCI   4           ;Charge la constante 4 dans la pile
                            ;arithmétique.
        CPLEI   fin         ;Détecte si l'introduction des données est
                            ;terminée. Vérifie si la constante 4 est
                            ;inférieure ou égale au registre entier
                            ;"point2". Le cas échéant, le programme
                            ;saute à "fin".
        SETRI   point1 400  ;Stocke la donnée 400 dans le registre
                            ;entier "point1".
        SETRI   point2 0    ;Stocke la donnée 0 dans le registre
                            ;entier "point2".
        JMP     fin         ;Saut inconditionnel à "fin"
fin     LD      r_in        ;Charge l'état du relais "r_in" dans la pile
                            ;logique.
        JNZ     fin1        ;Si le relais "r_in" est activé (Relais entrée),
                            ;le programme saute à "fin1". Dans le cas
                            ;contraire, l'exécution du programme se
                            ;poursuit.
        CLEAR               ;Efface la mémoire-tampon intermédiaire et
                            ;place le pointeur sur la première position.
        DISL    texte5      ;Copie, dans la mémoire-tampon
                            ;intermédiaire, le message de la table de
                            ;libellés "texte5".
        COM     0           ;Copie le contenu de la mémoire-tampon
                            ;intermédiaire sur le LCD.
fin1    END                 ;Fin du programme.
```

Certains équipements de la gamme MIDA disposent d'une mémoire pour bases de données de 64 Ko, dans laquelle il est possible de stocker au maximum 10 bases de données différentes.

Chacune d'entre elles (que nous appellerons fichier) est numérotée de 0 à 9. Toute opération relative à un fichier porte le numéro de celui-ci.

Les fichiers disposent de registres (fiches) dont le nombre est uniquement limité par la mémoire disponible. Chaque registre (fiche) est composé de champs. L'ensemble des registres pouvant comporter un nombre total maximum de 200 champs, si vous utilisez les 10 fichiers, chacun d'entre eux peut posséder 20 champs. Si vous utilisez 1 seul fichier, celui-ci peut comporter 200 champs.

Nous vous proposons, ci-après, un exemple d'organisation de la base de données de l'équipement.

Celle-ci peut comporter 2 fichiers (1 et 2) possédant respectivement 10 champs et 6 champs, et n registres :

BASE DE DONNÉES (maximum 10 fichiers, 200 champs pour l'ensemble des fichiers et 64 Ko de mémoire total utilisable)			
Registre (fiche)	**Fichier 1 (10 champs)**	**Fichier 2 (6 champs)**	**(*Maximum 10 fichiers)***
1			***(Maximum 200 champs)***
2			
3			
N			

Maximum 64 Ko

Les champs peuvent être de différents types :

TYPES DE CHAMPS			
LETTRE	**NOM**	**TYPE**	**TAILLE**
R	RELAIS	Relais	1 octet
I	INT	Entier	2 octets
F	FLOAT	Virgule flottante	4 octets
S	STRING	Texte	Variable
D	DATE	Date	3 octets
T	TIME	Heure	3 octets

La taille d'un champ de type "STRING" est le nombre de caractères indiqués dans la définition du fichier.

La taille d'un registre (fiche) se calcule en faisant la somme de la taille de chacun de ses champs (indiquée dans le tableau précédent).

La taille d'un fichier est la taille d'un registre multipliée par le nombre de registres (fiches) définis par l'intermédiaire de la pseudo-instruction de compilation FILE.

La lecture et l'écriture des fichiers s'effectue par registres complets grâce aux instructions READ et WRITE.

Le numéro du registre (fiche) sur lequel aura toujours lieu la lecture ou l'écriture (pointeur) est stocké dans des registres entiers (consultez le chapitre Adressage de la mémoire du Manuel Utilisateur de l'équipement correspondant). La valeur du pointeur peut être modifiée à volonté, ce qui permet d'exécuter la lecture et l'écriture de n'importe quel registre du fichier de la base de données.

Si vous tentez d'effectuer une opération sur un fichier non défini ou un numéro de registre inexistant, les commandes n'auront aucun effet et vous obtiendrez un message d'erreur.

✓ DÉFINITION DE LA STRUCTURE DES FICHIERS PSEUDO-INSTRUCTION "FILE"

Cette pseudo-instruction de compilation permet de définir la structure d'un fichier donné.

FILE N,C1,..,Cn,len=XXX,type=?

où :

N	Numéro du fichier (0 - 9).
C1,...**Cn**	Définition des champs.
Len = XXX	Longueur (XXX = nombre de registres).
Type	Type de fichier C/L (C = cyclique, L = linéaire).

N est le numéro du fichier (compris entre 0 et 9) et indique quel fichier vous souhaitez utiliser parmi les dix fichiers disponibles. Toute opération ayant trait à ce fichier sera référencée sous ce numéro.

Cn est la définition des champs du fichier. Ce paramètre se compose d'un chiffre et d'une lettre. Un fichier peut comporter au maximum 200 champs. La lettre définit le type de champ (voir tableau précédent) et le chiffre définit la destination (numéro de relais ou de registre). Vous trouverez une description des types de champs disponibles sur la page suivante.

Len = XXX est la longueur du fichier. Ce paramètre indique le nombre de registres qui composent le fichier. Ce chiffre peut être compris entre 1 et 65535, mais il est en réalité limité par la mémoire disponible qui dépend elle-même de l'espace occupé par les autres fichiers et de la taille du registre. Exemple : "len=1000" définit un fichier possédant 1000 registres (fiches).

Type indique si le fichier est de type cyclique ou linéaire. Si "type=c", le fichier retournera directement au premier registre après avoir rempli le dernier registre (fiche). Si "type=l", le fichier ne permettra pas de remplir d'autres registres (fiches) lorsque le dernier aura été rempli. Il indiquera une erreur, ce qui vous permettra d'éviter d'écraser des données. Par défaut, le fichier est de type linéaire ("type=l").

✓ TYPES DE CHAMPS DES REGISTRES

- **RELAIS (1 bit)**

Ces champs se définissent à l'aide de xxxR, où xxx est le numéro du relais sur lequel la valeur du champ RELAIS stockée sera lue ou écrite. Le numéro de relais est l'adresse de tout relais stocké dans la RAM et la RAM batterie.

- **INT (16 bits)**

Ces champs se définissent à l'aide de xxxI, où xxx est l'adresse du registre entier dans lequel la valeur du champ INT stockée sera lue ou écrite. Le numéro du champ entier est l'adresse de tout registre entier stocké dans la RAM batterie.

- **FLOAT (32 bits)**

Ces champs se définissent à l'aide de xxxF, où xxx est l'adresse du registre à virgule flottante dans lequel la valeur du champ FLOAT stockée sera lue ou écrite. Le numéro du champ à virgule flottante est l'adresse de tout registre à virgule flottante stocké dans la RAM batterie.

- **STRING (variable)**

Ces champs se définissent à l'aide de xxxS, où xxx est le nombre de caractères qui composent le texte.

Ce texte sera lu ou écrit sur la mémoire-tampon intermédiaire de l'équipement MIDA.

- **DATE (3 octets)**

Ces champs se définissent à l'aide de xxxD, où xxx est l'adresse de tout registre entier stocké dans la RAM batterie.

Pour lire la date figurant dans l'horloge interne, utilisez les registres entiers correspondants, décrits dans le chapitre Adressage de la mémoire du Manuel Utilisateur de l'équipement.

Pour écrire la date de l'horloge interne dans le fichier correspondant, il convient de savoir que les données sont stockées dans l'ordre ci-dessous :
XXX (date), XXX+1 (jour de la semaine), XXX+2 (mois), XXX+3 (année).

Si lors la définition du fichier, vous choisissez par exemple l'adresse du registre entier 500 en tant que XXXD, l'organisation des données se présentera de la manière suivante :

REGISTRE RAM	DONNÉE
500	Date
501	Jour de la semaine
502	Mois
503	Année

- **TIME (3 octets)**

Ces champs se définissent à l'aide de xxxT, où xxx est l'adresse de tout registre entier stocké dans la RAM batterie.

Pour lire l'heure figurant dans l'horloge interne, utilisez les registres entiers correspondants, décrits dans le chapitre Adressage de la mémoire du Manuel Utilisateur de l'équipement.

Pour écrire l'heure de l'horloge interne dans le fichier correspondant, il convient de savoir que les données sont stockées dans l'ordre ci-dessous :
XXX (secondes), XXX+1 (minutes) et XXX+2 (heures).

Si lors la définition du fichier, vous choisissez par exemple l'adresse du registre entier 600 en tant que XXXT, l'organisation des données se présentera de la manière suivante :

REGISTRE RAM	DONNÉE
600	Secondes
601	Minutes
602	Heures

✓ TAILLE DES FICHIERS

La capacité de stockage maximale des fichiers est de 64 Ko, c'est-à-dire 65536 octets. La taille d'un fichier est le produit de la taille d'un registre par le nombre de registres (fiches).

La taille d'un registre (fiche) se calcule en faisant la somme de la taille de chacun de ses champs (cette taille figure dans le tableau Types de champs fourni précédemment). La taille d'un champ de type "STRING" est le nombre de caractères indiqués dans la définition du fichier. Supposons que le fichier possède quatre champs : un relais ("RELAIS"), un entier ("INT"), une chaîne de 10 caractères ("STRING") et un champ heure ("TIME") et que sa définition soit la suivante :

FILE 1,400R,300I,10S,310T,len=1000,type=C

Type de champ	Taille
RELAIS	1
INT	2
STRING (10 caractères)	10
TIME	3
Total	16

Ce fichier possédant 1000 registres (fiches), sa taille est de 16000 octets, et il reste 49536 octets libres pour d'autres fichiers (65536 octets (64 Ko) - 16000 octets).

☑ Exemples de définition de fichiers :

A.- Définition du fichier 7 possédant les 4 champs suivants :

- 2 relais (600 et 601),
- 1 registre entier (400),
- 1 chaîne de caractères (9 caract.),
- l'heure (contenu du registre entier 700).

Étant donné que ce fichier comprend 500 registres et est de type cyclique, sa définition est la suivante :

```
FILE 7,600R,601R,400I,9S,700T,len=500,type=C
```

B.- Définition du fichier 3 possédant les 4 champs suivants :

- 2 registres à virgule flottante (20 et 55),
- la date (contenu du registre entier 400),
- l'heure (contenu du registre entier 410).

Étant donné que ce fichier comprend 1000 registres et est de type linéaire, sa définition est la suivante :

```
FILE 3,20F,55F,400D,410T,len=1000,type=L
```

READ

READ XXXX YYYY

MNÉMONIQUE : **READ**
OPÉRANDE XXXX : **Numéro du fichier de la base de données**
OPÉRANDE YYYY : **Constante pour le pointeur**
CODE INSTRUCTION : **38**

DESCRIPTION :

LIT les données d'un fichier de la base de données et les enregistre dans les registres (relais, registres entiers, registres à virgule flottante) spécifiés dans la définition (FILE) du fichier indiqué par l'opérande XXXX.
L'opérande XXXX est le numéro du fichier de la base de données à lire. La valeur autorisée est comprise entre 0 et 9.
L'opérande YYYY est la constante qui indique où placer le pointeur de lecture. La valeur de cette constante peut être +1, 0 ou -1 :

+1	Le pointeur du registre (fiche) doit être incrémenté après l'accès pour passer à la fiche suivante.
0	Le pointeur du registre (fiche) doit rester inchangé après l'accès.
-1	Le pointeur du registre (fiche) doit être décrémenté après l'accès pour passer à la fiche précédente.

Ne fait pas intervenir les piles (0).

☑ Exemple :

```
eti1    INK     340     ;Détection de la touche.
        JZ      eti1    ;Si la touche n'est pas activée, le
                        ;programme retourne à "éti1".
        READ    0   1   ;Lit, lors de chaque activation de la touche,
                        ;la fiche du fichier 0 de la base de données
                        ;dans laquelle le pointeur du fichier est
                        ;désigné. Incrémente le pointeur de 1 unité
                        ;pour la prochaine lecture.
        END
```

WRITE

WRITE XXXX YYYY

MNÉMONIQUE : **WRITE**
OPÉRANDE XXXX : **Numéro du fichier de la base de données**
OPÉRANDE YYYY : **Constante pour le pointeur**
CODE INSTRUCTION : **37**

DESCRIPTION :

ÉCRIT, dans le fichier de la base de données, le contenu des registres (relais, registres entiers, registres à virgule flottante) spécifiés dans la définition (FILE) du fichier indiqué par l'opérande XXXX.
L'opérande XXXX est le numéro du fichier de la base de données dans lequel les données doivent être écrites. La valeur autorisée est comprise entre 0 et 9.
L'opérande YYYY est la constante qui indique où placer le pointeur d'écriture. La valeur de cette constante peut être +1, 0 ou -1 :

- +1 Le pointeur du registre (fiche) doit être incrémenté après l'accès pour passer à la fiche suivante.
- 0 Le pointeur du registre (fiche) doit rester inchangé après l'accès.
- -1 Le pointeur du registre (fiche) doit être décrémenté après l'accès pour passer à la fiche précédente.

Ne fait pas intervenir les piles (0).

☑ Exemple :

```
eti1    INK       60     ;Détection de la touche.
        JZ        eti1   ;Attend que la touche soit activée. Si elle n'est pas
                         ;activée, le programme retourne à "éti1".
        WRITE     2   0  ;Écrit, lors de chaque activation de la touche, dans
                         ;la fiche du fichier 2 de la base de données dans
                         :laquelle le pointeur du fichier est désigné. Le
                         ;pointeur reste inchangé pour la prochaine
                         ;écriture.
        END
```

☑ EXEMPLES :

Avant d'exécuter ces exemples, vérifiez les instructions et l'adressage des relais/registres de mémoire disponibles dans l'équipement MIDA que vous vous disposez à programmer.

A.- Impression des données d'un fichier (MIDA 64) :

- Fichier nº 0 de type linéaire.
- Capacité : 1000 registres (fiches).
- Cinq champs : date, heure, code d'alarme (registre entier), température (registre à virgule flottante) et pression (registre à virgule flottante).
- Les données doivent être imprimées dans l'ordre suivant : des plus récentes au plus anciennes.
- Les données doivent être identifiées à l'aide de textes.

Définition du fichier :

File 0, 400D, 450T, 500I, 600F, 601F, Len=1000, Type=L

La taille du fichier ainsi défini est donc de :
(3 + 3 + 2 + 4 + 4) octets * 1000 registres = 16000 octets

Le programme est le suivant :

```
;
        File 0,400D,450T,500I,600F,601F,Len=1000,Type=L
;
;Définition des étiquettes.
expl        equ   399   ;Relais de 1ère exploration.
relcon      equ   500   ;Relais de contrôle d'écriture.
t_f1        equ   356   ;Touche <F1>.
t_f4        equ   359   ;Touche <F4>.
largeur     equ   20    ;Registre entier de largeur de page.
point_0     equ   50    ;Pointeur entier du fichier 0.

;Définition des libellés.
codal       lite    "CODE D'ALARME"
temp        lite    "TEMPÉRATURE = "
pres        lite    "PRESSION ="
```

```
oui         lite    "Messages"
non         lite    "Pas de message"

        LD          expl        ;Charge l'état du relais "expl" dans la pile
                                ;logique (il sera uniquement à "0" lors de la
                                ;première exploration du programme).
        JNZ         debut       ;Si l'état du relais "expl" est "1", le
                                ;programme saute à la ligne "début". Dans
                                ;le cas contraire, il passe à la ligne
                                ;suivante.
        MOVCI       80          ;Charge la constante 80 dans la pile
                                ;arithmétique.
        STOI        largeur     ;Stocke le contenu de la pile arithmétique
                                ;dans le registre "largeur" pour assigner
                                ;une largeur de ligne de 80 caractères sur
                                ;COM1.
debut   MOVRI       point_0     ;Charge le contenu du registre "point_0"
                                ;dans la pile arithmétique.
        MOVCI       -1          ;Charge la constante -1 dans la pile
                                ;arithmétique.
        CPEI        no_dt       ;Si le contenu du registre "point_0" est
                                ;égal à -1 (indique qu'il n'existe aucun
                                ;registre dans la base de données), le
                                ;programme saute à la ligne "no_dt". Dans
                                ;le cas contraire, il passe à la ligne
                                ;suivante.
        CLEAR                   ;Efface la mémoire-tampon intermédiaire.
        DISL        si          ;Copie le texte "oui" de la table des libellés
                                ;dans la mémoire-tampon intermédiaire.
        COM         0           ;Copie le contenu de la mémoire-tampon
                                ;intermédiaire sur le LCD.
        JMP         writs       ;Saute à la line "writs".
no_dt   CLEAR                   ;Efface la mémoire-tampon intermédiaire.
        DISL        non         ;Copie le texte "non" de la table des libellés
                                ;dans la mémoire-tampon intermédiaire.
        COM         0           ;Copie le contenu de la mémoire-tampon
                                ;intermédiaire sur le LCD.
writs   INK         t_f4        ;Détection de la touche <F4>.
```

```
        JZ          reads       ;Si la touche <F4> n'est pas activée, le
                                ;programme saute à la ligne "reads". Dans
                                ;le cas contraire, il passe à la ligne
                                ;suivante.
        MOVRI       point_0     ;Charge le contenu du registre "point_0"
                                ;dans la pile arithmétique.
        MOVCI       -1          ;Charge la constante -1 dans la pile
                                ;arithmétique.
        CPGI        writ1       ;Si le contenu du registre "point_0" est
                                ;supérieur à -1, le programme saute à la
                                ;ligne "writ1". Dans le cas contraire, il passe
                                ;à la ligne suivante.
        INC         point_0 1   ;Incrémente de 1 niveau le contenu du
                                ;registre "point_0" (place le pointeur sur le
                                ;premier registre).
writ1   WRITE       0       1   ;Écrit dans le fichier 0 de la base de
                                ;données et incrémente le pointeur de
                                ;1 niveau (registre "point_0") chaque fois
                                ;que la touche <F4> est activée.
        SET         relcon      ;Active le relais "relcon" (indiquant qu'une
                                ;écriture a eu lieu).
reads   INK         t_f1        ;Détection de la touche <F1>.
        JZ          fin         ;Si la touche <F1> n'est pas activée, le
                                ;programme saute à la ligne "fin". Dans le
                                ;cas contraire, il passe à la ligne suivante.
        MOVRI       point_0     ;Charge le contenu du registre "point_0"
                                ;dans la pile arithmétique.
        MOVCI       -1          ;Charge la constante -1 dans la pile
                                ;arithmétique.
        CPLEI       fin         ;Si le contenu du registre "point_0" est
                                ;inférieur ou égal à -1, le programme
                                ;saute à la ligne "fin" (il n'y a pas de
                                ;registres dans la base de données). Dans
                                ;le cas contraire, il passe à la ligne
                                ;suivante.
        LD          relcon      ;Charge l'état du relais "relcon".
        JZ          read1       ;Si l'état du relais "relcon" est "0", le
                                ;programme saute à la ligne "read1".
```

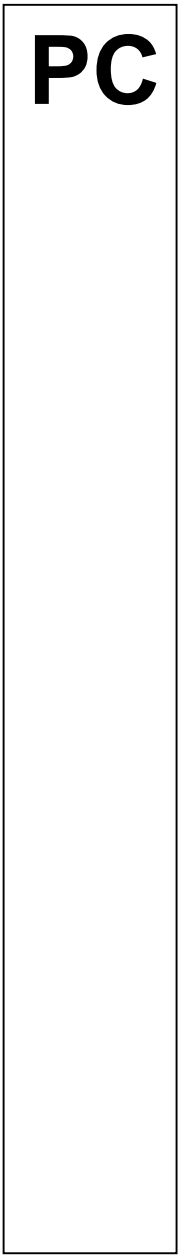
```
        INC     point_0 -1  ;Décrémente de 1 niveau le contenu du
                            ;registre "point_0" (place le pointeur sur le
                            ;dernier registre écrit dans la base de
                            ;données).
read1   MOVRI   point_0     ;Charge le contenu du registre "point_0"
                            ;dans la pile arithmétique.
        MOVCI   -1          ;Charge la constante -1 dans la pile
                            ;arithmétique.
        CPGI    read2       ;Si le contenu du registre "point_0" est
                            ;supérieur à -1, le programme saute à la
                            ;ligne "read2" (il existe des registres dans
                            ;la base de données). Dans le cas
                            ;contraire, il passe à la ligne suivante.
        JMP     fin         ;Saute à la ligne "fin".
read2   READ    0       -1  ;Écrit dans le fichier 0 de la base de
                            ;données et décrémente le pointeur de
                            ;1 niveau (registre "point_0").
        CLEAR               ;Efface la mémoire-tampon intermédiaire.
        DISRI   400     2   ;Copie le contenu du registre 400 (jour)
                            ;dans la mémoire-tampon.
        DISCH   47          ;Copie le caractère ASCII 47 (/) dans la
                            ;mémoire-tampon.
        DISRI   402     2   ;Copie le contenu du registre 402 (mois)
                            ;dans la mémoire-tampon.
        DISCH   47          ;Copie le caractère ASCII 47 (/) dans la
                            ;mémoire-tampon.
        DISRI   403     2   ;Copie le contenu du registre 403 (année)
                            ;dans la mémoire-tampon.
        DISCH   32          ;Copie le caractère ASCII 32 (espace en
                            ;blanc) dans la mémoire-tampon.
        DISRI   452     2   ;Copie le contenu du registre 452 (heure)
                            ;dans la mémoire-tampon.
        DISCH   58          ;Copie le caractère ASCII 58 (:) dans la
                            ;mémoire-tampon.
        DISRI   451     2   ;Copie le contenu du registre 451 (minutes)
                            ;dans la mémoire-tampon.
        DISCH   32          ;Copie le caractère ASCII 32 (espace en
                            ;blanc) dans la mémoire-tampon.
```

```
        DISL    codal       ;Copie le texte "codal" de la table des
                            ;libellés dans la mémoire-tampon
                            ;intermédiaire.
        DISRI   500   3     ;Copie le contenu du registre entier 500
                            ;dans la mémoire-tampon.
        DISCH   13          ;Copie le caractère ASCII 13 (saut de
                            ;ligne) dans la mémoire-tampon.
        DISL    temp        ;Copie le texte "temp" de la table des
                            ;libellés dans la mémoire-tampon
                            ;intermédiaire.
        DISRF   600   51    ;Copie le contenu du registre à virgule
                            ;flottante 600 dans la mémoire-tampon.
        DISCH   32          ;Copie le caractère ASCII 32 (espace en
                            ;blanc) dans la mémoire-tampon.
        DISL    pres        ;Copie le texte "pres" de la table des
                            ;libellés dans la mémoire-tampon
                            ;intermédiaire.
        DISRF   601   40    ;Copie le contenu du registre à virgule
                            ;flottante 601 dans la mémoire-tampon.
        COM     1           ;Transmet, via COM1 (RS232), le contenu
                            ;de la mémoire-tampon intermédiaire, qui
                            ;est une ligne de détection d'alarme avec
                            ;les lectures de température et de pression
                            ;d'un système.
        RESET   relcon      ;Désactive le relais "relcon".
fin     END                 ;Fin du programme.
```

Le résultat imprimé de l'exemple précédent est :

```
JJ/MM/AA HH:MM CODE D'ALARME XXX
TEMPÉRATURE = XXX.X PRESSION = XXXX
```

où les "X" sont des données extraites du fichier de la base de données.


PC

AFEISA a créé pour les équipements MIDA un protocole de communication qui leur est propre.

Les principales caractéristiques de ce protocole sont les suivantes :

- Il est de type maître-esclave avec confirmation.
- Les communications ont lieu caractère par caractère en HEXADÉCIMAL-ASCII.
- Tous les messages acceptables transmis à l'équipement font l'objet d'une confirmation ou d'une réponse.
- Si le message est inintelligible ou s'il est impossible de déterminer le numéro de périphérique correspondant (erreur de communication, message incorrect, etc.), l'équipement ne répond pas.
- Le numéro de périphérique étant correct, si le message présente un défaut de forme, l'équipement répond en transmettant un message d'erreur.
- Tous les messages de réponse de l'équipement comportent un octet d'état dans lequel se reflètent d'éventuelles erreurs de communication et d'exécution de programme.
- Les messages envoyés au périphérique 00 par le PC sont reçus par tous les équipements en réseau, mais aucun d'entre eux ne répond. Les messages commencent par un caractère '/' et se terminent par un caractère ';'.

Tous les équipements de la nouvelle gamme MIDA disposent de ce protocole et peuvent être connectés en réseau RS485 et RS422.

LISTE DES FONCTIONS DU PROTOCOLE MIDAbus

Le protocole de communication MIDAbus prend en charge les fonctions suivantes :

MESSAGE	FONCTION
Message 10	Demande concernant la version de l'équipement.
Message 11	Commande d'initialisation de l'équipement.
Message 13	Demande concernant l'état (Marche/Arrêt) de l'équipement.
Message 93	Modification de l'état (Marche/Arrêt) de l'équipement.
Message 91	Commande d'effacement de registres.
Message 15	Demande concernant l'état des relais par groupes de huit.
Message 95	Modification de l'état des relais par groupes de huit.
Message 16	Demande concernant l'état des relais.
Message 96	Modification de l'état des relais.
Message 17	Demande concernant le contenu de registres entiers.
Message 97	Modification du contenu de registres entiers.
Message 1B	Demande concernant le contenu de registres de 32 bits à virgule flottante.
Message 9B	Modification du contenu de registres de 32 bits à virgule flottante.
Message 18	Demande concernant le contenu des écrans.
Message 98	Modification du contenu des écrans.
Message 9C	Simulation d'activation des touches.
Message 1D	Demande concernant le contenu de l'horloge interne de l'équipement.
Message 9D	Modification du contenu de l'horloge interne de l'équipement (mise à l'heure).

Message 20	Demande concernant le format d'un fichier.
Message 21	Demande concernant le format des champs d'un fichier.
Message 22	Demande concernant un bloc mémoire des fichiers.
Message A2	Enregistrement d'un bloc mémoire de fichiers.

FORMAT GÉNÉRAL DES MESSAGES

Chaque bloc de deux caractères, séparés ou non par un espace, représente un octet. Exemple :

NP CC IIII... = 4 octets

Le format général des messages est le suivant :

Début	Périph.	Code	Adr.	Donnée	État	Check	Fin
1 octet	1 octet	1 octet	2 octets	120 octets	1 octet	1 octet	1 octet
/	NP	CC	IIII	DDD	ST	CK	;

/ NP CC IIII DDDD... ST CK ;

/ ..... Caractère de début de message.
NP ..... Numéro de périphérique.
CC ..... Code du message (type d'opération).
IIII ..... Adresse des demandes ou des modifications.
DDDD ..... Données du message (plusieurs octets).
ST ..... État de l'équipement.
CK ..... Total de contrôle (Checksum) du message.
; ..... Caractère de fin de message.

Le message décrit peut varier en fonction du type de message (de demande ou de réponse).

CALCUL DU CHECK-SUM

Le total de contrôle (Checksum) s'obtient en calculant, octet par octet, en hexadécimal, la somme de tous les composants du message, excepté les caractères de début et de fin.

Le total de contrôle s'exprime sous la forme de deux caractères (1 octet) en hexadécimal. Si le résultat obtenu occupe deux octets, le plus significatif est ignoré.

☑ **Exemple :**

Message **/051700640ACK;**

CK = 05 + 17 + 00 + 64 + 0A = 8A

RÉSULTAT..... /051700640A8A;

☑ **Exemple :**

Message **/150704830064CK;**

CK = 15 + 07 + 04 + 83 + 00 + 64 = 107 = 07

RÉSULTAT.... /15070483006407;

TAILLE MAXIMALE DES MESSAGES

La mémoire-tampon intermédiaire des équipements MIDA varie selon le modèle (consultez le Manuel Utilisateur correspondant).
Tenez compte du fait que la longueur d'un message ne peut en aucun cas être supérieure à celle admise par la mémoire-tampon intermédiaire (caractères de début et de fin de message compris).

OCTET D'ÉTAT

Chaque réponse du MIDA implique la transmission d'un octet qui indique l'état de l'équipement.
Si cet octet est "00", l'équipement ne présente ni problème, ni erreur, mais si l'un des bits est activé, cela traduit un problème de l'équipement ou une erreur de programme.
L'octet d'état est codé de la manière suivante :

BIT	ERREUR
Bit 0 activé (1 Hex)	Message impossible à traiter lorsque l'équipement est en marche.
Bit 1 activé (2 Hex)	Plage de variables demandées incorrecte.
Bit 2 activé (4 Hex)	Erreur due à une tentative de lecture ou d'écriture en dehors des limites de la base de données.
Bit 3 activé (8 Hex)	Erreur due à une tentative de lecture ou d'écriture en dehors des limites fixées par la topographie de la mémoire.
Bit 4 activé (10 Hex)	Erreur due à un dépassement mathématique ou à une tentative de division par zéro.
Bit 5 activé (20 Hex)	Erreur dans l'une des piles (logique, arithmétique ou de sous-routines).
Bit 6 activé (40 Hex)	Erreur de check-sum de FLASH-EPROM.
Bit 7 activé (80 Hex)	Erreur de lecture ou d'écriture d'EEPROM.

Consultez le chapitre "Traitement des erreurs" de ce manuel.

MESSAGE 10 :
DEMANDE CONCERNANT LA VERSION DE L'ÉQUIPEMENT

Message 10	
Question	Réponse
/NP 10 CK;	Rép. OK /NP 90 HH SS ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
HH ..... Version du matériel (Hardware).
SS ..... Version du logiciel (Software).
ST ..... État de l'équipement.
CK ..... Check-sum du message.

☑ **Exemple :** Demande de version à l'unité correspondant au périphérique numéro 15.

MESSAGE PC -> MIDA : **/151025;**

15 Numéro de périphérique (15).
10 Code du message.
25 Check-sum du message.

MESSAGE MIDA -> PC : **/1590202000E5;**

15 Numéro de périphérique (15).
90 Réponse au message 10.
20 Version du matériel (Hardware) (20).
20 Version du logiciel (Software) (20 ⇨ v2.0).
00 Octet d'état (absence d'erreur).
E5 Check-sum du message.

MESSAGE 11 :
COMMANDE D'INITIALISATION DE L'ÉQUIPEMENT

Message 11	
Question	Réponse
/NP 11 DD CK;	Rép. OK Pas de réponse. Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
DD ..... Données
FF = RÉINITIALISATION HARDWARE.
00 = RÉINITIALISATION SOFTWARE.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : le MIDA réinitialise la partie matérielle de l'équipement (Hardware) à l'aide du chien de garde de celui-ci.

☑ **Exemple :** Demande de réinitialisation du Hardware de l'unité 15.

MESSAGE PC -> MIDA : **/1511FF25;**

15 Numéro de périphérique (15).
11 Code du message.
FF Demande de réinitialisation du Hardware.
25 Check-sum du message.

MESSAGE MIDA -> PC : **PAS DE RÉPONSE**

MESSAGE 13 :
DEMANDE CONCERNANT L'ÉTAT (MARCHE/ARRÊT) DE L'ÉQUIPEMENT

Message 13	
Question	Réponse
/NP 13 CK;	Rép. OK /NP 93 DD ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
DD ..... Données
FF = Équipement en MARCHE.
00 = Équipement à l'ARRÊT.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

☑ **Exemple :** Demande d'état à l'unité 15.

MESSAGE PC -> MIDA : **/151328;**

15 Numéro de périphérique (15).
13 Code du message.
28 Check-sum du message.

MESSAGE MIDA -> PC : **/1593FF00A7;**

15 Numéro de périphérique (15).
93 Réponse au message 13.
FF État de l'équipement (marche).
00 Octet d'état (absence d'erreur).
A7 Check-sum du message.

MESSAGE 93 :
MODIFICATION DE L'ÉTAT (MARCHE/ARRÊT) DE L'ÉQUIPEMENT

Message 93	
Question	Réponse
/NP 93 DD CK;	Rép. OK /NP 00 ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
DD ..... Données
FF = Équipement en MARCHE.
00 = Équipement à l'ARRÊT.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

☑ **Exemple :** Demande de modification d'état à l'unité 15.

MESSAGE PC -> MIDA : **/159300A8;**

15 Numéro de périphérique (15).
93 Code du message.
00 Modification de l'état de l'équipement (ARRÊT).
A8 Check-sum du message.

MESSAGE MIDA -> PC : **/15000015;**

15 Numéro de périphérique (15).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
15 Check-sum du message.

MESSAGE 91 :
COMMANDE D'EFFACEMENT DE REGISTRES

Message 91	
Question	Réponse
/NP 91 DD CK;	Rép. OK /NP 00 ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.

DD ..... Données
- 00 = Effacement du contenu des registres RAM, EEPROM.
- 01 = Effacement du contenu de la base de données.
- 02 = Exécution des commandes 00 et 01.
- 03 = Exécution des commandes 00, 01 et récupération de la CONFIGURATION par défaut de l'équipement.

ST ..... État de l'équipement.

CK ..... Check-sum du message.

REMARQUE : Pour que tous les changements soient effectifs, il est nécessaire d'envoyer un message d'initialisation (Message 11) ou d'éteindre puis de rallumer l'équipement.

☑ **Exemple :** Effacement RAM et EEPROM de l'équipement 1.

MESSAGE PC -> MIDA : **/01910092;**

01 Numéro de périphérique (1).
91 Code du message.
00 Effacement du contenu des registres RAM et EEPROM.
92 Check-sum du message.

MESSAGE MIDA -> PC : **/01000001;**

01 Numéro de périphérique (1).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
01 Check-sum du message.

MESSAGE 15 : DEMANDE CONCERNANT L'ÉTAT DES RELAIS PAR GROUPES DE HUIT

Message 15	
Question	Réponse
/NP 15 II NN CK;	Rép. OK /NP 95 II DD ... ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
II ..... Numéro du premier bloc objet de la demande.
(Relais 0 - 7 = Bloc 0)
(Relais 8 - 15 = Bloc 1)
NN ..... Nombre de blocs objets de la demande.
DD ..... Blocs réponse (8 relais par octet).
Code :
x x x x | x x x x = Bits du bloc
1 2 3 4 5 6 7 8 = Relais objets de la demande.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : La demande peut se référer à un nombre maximum de 60 blocs.

Exemple : Demande concernant les deux premiers blocs de relais de l'unité 1 (entrées 0–15).

MESSAGE PC -> MIDA : **/0115000218;**

01 Numéro de périphérique (1).
15 Code du message.
00 Premier bloc objet de la demande (bloc 0).
02 Nombre de blocs objets de la demande (2 blocs, entrées 0-15).
18 Check-sum du message.

MESSAGE MIDA -> PC : **/019500C1AB0002;**

01 Numéro de périphérique (1).
95 Réponse au message 15.
00 Bloc de début de la réponse (bloc 0).
C1 État du premier bloc de la réponse (bloc 0).
Relais 0 à relais 7 (1 1 0 0 0 0 0 1).
AB État du deuxième bloc de la réponse (bloc 1).
Relais 8 à relais 15 (1 0 1 0 1 0 1 1).
00 Octet d'état (absence d'erreur).
02 Check-sum du message.

MESSAGE 95 :
MODIFICATION DE L'ÉTAT DES RELAIS PAR GROUPES DE HUIT

Message 95	
Question	Réponse
/NP 95 II DD CK;	Rép. OK /NP 95 00 ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
II ..... Numéro du premier bloc objet de la demande.
(Relais 0 - 7 = Bloc 0)
(Relais 8 - 15 = Bloc 1)
DD ..... Blocs réponse (8 relais par octet).
Code :
x x x x | x x x x = Bits du bloc
1 2 3 4 5 6 7 8 = Relais objet de la demande.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : La modification peut porter sur un nombre maximum de 60 blocs.

☑ **Exemple :** Modification des blocs de relais 12 et 13 de l'unité 1 :

Bloc 12 (96 - 103) 1 1 1 1 1 1 1 1 = FF
Bloc 13 (104 - 111) 0 0 1 1 1 1 1 1 = 3F

MESSAGE PC -> MIDA : **/01950CFF3FE0;**

01 Numéro de périphérique (1).
95 Code du message.
0C Bloc du début de la modification (bloc 12).

FF Modification du premier bloc (bloc 12).
3F Modification du deuxième bloc (bloc 13).
E0 Check-sum du message.

MESSAGE MIDA -> PC : **/01000001;**

01 Numéro de périphérique (1).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
01 Check-sum du message.

MESSAGE 16 :
DEMANDE CONCERNANT L'ÉTAT DES RELAIS

Message 16	
Question	Réponse
/NP 16 IIII NN CK;	Rép. OK /NP 96 IIII DD ... ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
IIII ..... Adresse du premier relais objet de la demande.
NN ..... Nombre de relais objets de la demande.
DD ..... Données de réponse (1 relais par octet).
00 = Relais DÉSACTIVÉ
FF = Relais ACTIVÉ
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : La demande peut se référer à un nombre maximum de 60 relais.

☑ **Exemple :** Demande concernant 6 relais de l'unité 1 (à partir du relais 225).

MESSAGE PC -> MIDA : **/011600FF061C;**

01 Numéro de périphérique (1).
16 Code du message.
00FF Adresse du premier relais objet de la demande (relais 255).
06 Nombre de relais objets de la demande (6).
1C Check-sum du message.

MESSAGE MIDA -> PC : **/019600FFFFFF000000000094;**

01 Numéro de périphérique (1).
96 Réponse au message 16.
00FF Adresse du premier relais objet de la demande (relais 255).
FF État du premier relais objet de la demande (relais 255 ON).
FF État du deuxième relais objet de la demande (relais 256 ON).
00 État du troisième relais objet de la demande (relais 257 OFF)
00 État du quatrième relais objet de la demande (relais 258 OFF)
00 État du cinquième relais objet de la demande (relais 259 OFF)
00 État du sixième relais objet de la demande (relais 260 OFF)
00 Octet d'état (absence d'erreur).
94 Check-sum du message.

MESSAGE 96 :
MODIFICATION DE L'ÉTAT DES RELAIS

Message 96	
Question	Réponse
/NP 96 IIII DD ... CK;	Rép. OK /NP 96 00 ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
IIII ..... Adresse du premier relais objet de la modification.
DD ..... Données de modification (1 octet par relais).
00 = Relais DÉSACTIVÉ
FF = Relais ACTIVÉ
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : La modification peut porter sur un nombre maximum de 60 relais.

☑ **Exemple :** Activation du relais 8 et désactivation du relais 9 de l'unité 7.

MESSAGE PC -> MIDA : **/07960008FF00A4;**
07 Numéro de périphérique (7).
96 Code du message.
0008 Adresse du premier relais objet de la modification (relais 8).
FF Activation du premier relais (relais 8 ON).
00 Désactivation du second relais (relais 9 OFF).
A4 Check-sum du message.

MESSAGE MIDA -> PC : **/07000007;**
07 Numéro de périphérique (7).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
07 Check-sum du message.

MESSAGE 17 :
DEMANDE CONCERNANT LE CONTENU DE REGISTRES ENTIERS

Message 17	
Question	Réponse
/NP 17 IIII NN CK;	Rép. OK /NP 97 IIII DD... ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
IIII ..... Adresse du premier registre objet de la demande.
NN ..... Nombre de registres objets de la demande.
DD ..... Données de la réponse (2 octets par registre).
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : La demande peut se référer à un nombre maximum de 60 registres entiers.

Exemple : Demande concernant 3 registres entiers à partir de l'adresse 4 de l'unité 1.

MESSAGE PC -> MIDA : **/011700FA0315;**

01 Numéro de périphérique (1).
17 Code du message.
00FA Adresse du premier registre objet de la demande (registre 250)
03 Nombre de registres entiers objets de la demande (3).
15 Check-sum du message.

16 MESSAGE MIDA -> PC : **/019700FAD8F002D33A980001;**

01 Numéro de périphérique (1).
97 Réponse au message 17.
00FA Adresse du premier registre objet de la demande (250).
D8F0 Contenu du premier registre (registre 250 : –22769).
02D3 Contenu du deuxième registre (registre 251 : 723).
3A98 Contenu du troisième registre (registre 252 : 15000).
00 Octet d'état (absence d'erreur).
01 Check-sum du message.

MESSAGE 97 :
MODIFICATION DU CONTENU DE REGISTRES ENTIERS

Message 97	
Question	Réponse
/NP 97 IIII DD ... CK;	Rép. OK /NP 97 00 ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
IIII ..... Adresse du premier registre objet de la modification.
DD ..... Données de la modification (2 octets par registre).
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : La modification peut porter sur un nombre maximum de 30 registres.

☑ **Exemple :** Modification des registres 100 et 101 de l'unité 1.

MESSAGE PC -> MIDA : **/019701F406BB0065B3;**

01 Numéro de périphérique (1).
97 Code du message.
01F4 Adresse du premier registre objet de la modification (reg. 500).
06BB Modification du premier registre (registre 500 : 1723).
0065 Modification du deuxième registre (registre 501 : 101).
B3 Check-sum du message.

MESSAGE MIDA -> PC : **/01000001;**

01 Numéro de périphérique (1).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
01 Check-sum du message.

MESSAGE 1B : DEMANDE CONCERNANT LE CONTENU DE REGISTRES DE 32 BITS À VIRGULE FLOTTANTE

Message 1B	
Question	Réponse
/NP 1B IIII NN CK;	Rép. OK /NP 9B IIII DD... ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
IIII ..... Adresse du premier registre objet de la demande.
NN ..... Nombre de registres objets de la demande.
DD ..... Données de la réponse (4 octets par registre).
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : La demande peut se référer à un nombre maximum de 15 registres à virgule flottante.
Format des données : 32 bits, hexadécimal (IEEE).

☑ **Exemple :** Demande concernant le contenu de 3 registres à virgule flottante à partir de l'adresse 4 de l'unité 1.

MESSAGE PC -> MIDA : **/011B00040323;**

01 Numéro de périphérique (1).
1B Code du message.
0004 Adresse du premier registre objet de la demande (registre 4).
03 Nombre de registres à virgule flottante objets de la demande (3).
23 Check-sum du message.

MESSAGE MIDA -> PC :

/019B0004C61C40004434E000466A6000002A;

01	Numéro de périphérique (1).
9B	Réponse au message 1B.
0004	Adresse du premier registre objet de la demande (reg. 4).
C61C4000	Contenu du premier registre (registre 4 : -10000).
4434E000	Contenu du deuxième registre (registre 5 : 723.5).
466A6000	Contenu du troisième registre (registre 6 : 15000).
00	Octet d'état (absence d'erreur).
2A	Check-sum du message.

MESSAGE 9B :
MODIFICATION DU CONTENU DE REGISTRES DE 32 BITS À VIRGULE FLOTTANTE

Message 9B	
Question	Réponse
/NP 9B IIII DD... CK;	Rép. OK /NP 00 ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
IIII ..... Adresse du premier registre objet de la modification.
DD ..... Données de la réponse (4 octets par registre).
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : La modification peut porter sur un nombre maximum de 15 registres à virgule flottante.
Format des données : 32 bits, hexadécimal (IEEE).

☑ **Exemple :** Modification des registres 100 et 101 de l'unité 1.

MESSAGE PC -> MIDA : **/019B006443E000000C7F1200016;**

01	Numéro de périphérique (1).
9B	Code du message.
0064	Adresse du premier registre objet de la modification (registre 100).
3E000000	Modification du premier registre (registre 100 : 0.125)
C7F12000	Modification du deuxième registre (registre 101 : -123456)
17	Check-sum du message.

MESSAGE MIDA -> PC : **/01000001;**

01 Numéro de périphérique (1).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
01 Check-sum du message.

MESSAGE 18 :
DEMANDE CONCERNANT LE CONTENU DES ÉCRANS

Message 18	
Question	Réponse
/NP 18 *LL* CK;	Rép. OK /NP 98 *LL* DD... ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
LL ..... Lignes du LCD :
- 00 - 1ère ligne du LCD
- 01 - 2ème ligne du LCD
- 02 - 3ème ligne du LCD
- 03 - 4ème ligne du LCD
- 04 - Écran rouge

DD ..... Données de la réponse (20 octets en ASCII -HEXA).
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : L'option *LL* est exclusivement disponible sur les équipements dotés d'un LCD à 4 lignes. Elle n'est pas utilisable sur les autres équipements.
Sans l'option *LL*, les 32 premiers caractères de la réponse correspondent au LCD et les 12 autres à l'écran rouge.
Avec l'option *LL*, chaque ligne dispose de 20 caractères (12 pour l'écran rouge).

☑ **Exemple :** Demande concernant le contenu des écrans de l'unité 1 (sans option *LL*).

MESSAGE PC -> MIDA : **/011819;**
01 Numéro de périphérique (1).
18 Code du message.
19 Check-sum du message.

MESSAGE MIDA -> PC :

/0198454A45435554494F4E2020202020202050524F4752414D4D452E2E2E202020202D313135303020202020202000B7;

01 Numéro de périphérique (1).
98 Réponse au message 18.
454A45435554494F4E20202020202020 Première ligne du LCD "EXECUTION"
50524F4752414D4D452E2E2E20202020 Deuxième ligne du LCD "PROGRAMME..."
2D3131353030202020202020 Contenu de l'écran rouge. "-11500"
00 Octet d'état (absence d'erreur).
B7 Check-sum du message.

☑ **Exemple :** Demande concernant le contenu de la 3ème ligne du LCD de l'unité 2 (avec option *LL*).

MESSAGE PC -> MIDA : **/0218021D;**

02 Numéro de périphérique (2).
18 Code du message.
02 Numéro de la ligne du LCD objet de la demande (3).
1D Check-sum du message.

MESSAGE MIDA -> PC :

/0298022045544154204 14C41524D45203A204F464620200051;

02 Numéro de périphérique (2).
98 Réponse au message 18.
02 Demande concernant la 3ème ligne du LCD
204554415420414C41524D45203A204F46462020
Contenu de la 3ème ligne de l'écran : " Etat alarme :OFF "
00 Octet d'état (absence d'erreur).
51 Check-sum du message.

MESSAGE 98 :
MODIFICATION DU CONTENU DES ÉCRANS

Message 98	
Question	Réponse
/NP 98 *LL* CK;	Rép. OK /NP 00 ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
LL ..... Lignes du LCD :
- 05 - 1ère ligne du LCD
- 06 - 2ème ligne du LCD
- 07 - 3ème ligne du LCD
- 08 - 4ème ligne du LCD
- 09 - Écran rouge

DD ..... Données de la réponse (20 octets en ASCII -HEXA).
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : L'option *LL* est exclusivement disponible sur les équipements dotés d'un LCD à 4 lignes. Elle n'est pas utilisable sur les autres équipements.
Sans l'option *LL*, les 32 premiers caractères de la réponse correspondent au LCD et les 12 autres à l'écran rouge.
Avec l'option *LL*, chaque ligne dispose de 20 caractères (12 pour l'écran rouge).

☑ **Exemple :** Modification du contenu du LCD
"NUMERO FORMULE ? +***" et de l'écran rouge
("0.0") dans l'unité 1 (sans option *LL*).

MESSAGE MIDA -> PC :

**/01984E554D45524F20464F524D554C45203F202020202020
2B2A2A2A20202020202020202020302E30202020202015;**

01 Numéro de périphérique (1).
98 Réponse au message 18.
4E554D45524F20464F524D554C45203F Première ligne du LCD : "NUMÉRO FORMULE ?"
2020202020202B2A2A2A202020202020 Deuxième ligne du LCD: "+***"
2D313135303020202020202020 Contenu de l'écran rouge : "0.0"
15 Check-sum du message.

MESSAGE MIDA -> PC : **/01000001;**

01 Numéro de périphérique (1).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
01 Check-sum du message.

☑ **Exemple :** Modification du contenu de la 1ère ligne du LCD (PRIORITÉ 001) de l'unité 2 (avec option *LL*).

MESSAGE PC -> MIDA :
/029800205052494F52495445203030312020202020202000B9;

02 Numéro de périphérique (2).
98 Code du message.
00 Numéro de la ligne du LCD objet de la modification (1).
205052494F52495445203030312020202020202020
Modification du LCD : "PRIORITÉ 001"
B9 Check-sum du message.

MESSAGE MIDA -> PC : **/02000002;**

02 Numéro de périphérique (2).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
02 Check-sum du message.

MESSAGE 9C :
SIMULATION DE L'ACTIVATION DE TOUCHES

Message 9C	
Question	Réponse
/NP 9C TTTT CK;	Rép. OK⇨ /NP 00 ST CK; Rép. NOK⇨/NP FF ST CK;

où :

NP ..... Numéro de périphérique.
TTTT ..... Code de la touche (relais interne selon modèle).
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : Les codes des touches correspondent aux relais internes de chacune d'entre elles (consultez la topographie de la mémoire dans le Manuel Utilisateur de l'équipement MIDA correspondant).

☑ **Exemple :** Simulation de l'activation de la touche <ENTER> de l'unité 1 (ex. : MIDA 20).

MESSAGE PC -> MIDA : **/019C003CD9;**

01 Numéro de périphérique (1).
9C Code du message.
003C Code de la touche <ENTER>.
D9 Check-sum du message.

MESSAGE MIDA -> PC : **/01000001;**

01 Numéro de périphérique (1).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
01 Check-sum du message.

MESSAGE 1D :
DEMANDE CONCERNANT LE CONTENU DE L'HORLOGE INTERNE DE L'ÉQUIPEMENT

Message 1D	
Question	Réponse
/NP 1D CK;	Rép. OK /NP 9D SS MM HH DD mm AA WW ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
SS ..... Secondes.
MM ..... Minutes.
HH ..... Heures.
DD ..... Date.
mm ..... Mois.
AA ..... Année.
WW ..... Jour de la semaine.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : Les données sont transmises au format décimal (réel).

☑ **Exemple :** Demande concernant le contenu de l'horloge interne de l'unité 1.

MESSAGE PC -> MIDA : **/011D1E;**

01 Numéro de périphérique (1).
1D Code du message.
1E Check-sum du message.

MESSAGE MIDA -> PC : **/019D0030122804950400A5;**

01 Numéro de périphérique (1).
9D Réponse au message 1D.
00 Secondes écoulées (0 s).
30 Minutes écoulées (30 min).
12 Heure (12h00).
28 Jour du mois (28).
04 Mois de l'année (avril).
95 Année (1995).
04 Jour de la semaine (vendredi).
00 Octet d'état (absence d'erreur).
A5 Check-sum du message.

MESSAGE 9D :
MODIFICATION DU CONTENU DE L'HORLOGE INTERNE DE L'ÉQUIPEMENT (MISE À L'HEURE)

Message 9D	
Question	Réponse
/NP 9D SS MM HH DD mm AA WW CK;	Rép. OK /NP 00 ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
SS ..... Secondes.
MM ..... Minutes.
HH ..... Heures.
DD ..... Date.
mm ..... Mois.
AA ..... Année.
WW ..... Jour de la semaine.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : Les données sont transmises au format décimal (réel).

☑ **Exemple :** Mise à l'heure de l'horloge interne.

MESSAGE PC -> MIDA : **/019D00450029049505AA;**
01 Numéro de périphérique (1).
01 Numéro de périphérique (1).
9D Code du message.
00 Rectification des secondes écoulées (0 s).
45 Rectification des minutes écoulées (45 min).
00 Rectification de l'heure (00h00).
29 Rectification de la date (29).
04 Rectification du mois (avril).

95 Rectification de l'année (1995).
05 Rectification du jour de la semaine (samedi).
AA Check-sum du message.

MESSAGE MIDA -> PC : **/01000001;**

01 Numéro de périphérique (1).
00 Confirmation de la modification OK.
00 Octet d'état (absence d'erreur).
01 Check-sum du message.

MESSAGE 20 :
DEMANDE CONCERNANT LE FORMAT D'UN FICHIER

Message 20	
Question	Réponse
/NP 20 nn CK;	Rép. OK /NP A0 pppp rrrr nnnn tt cc dd xx ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
nn ..... Numéro de fichier (0-9).
pppp ..... Offset fichier.
rrrr ..... Taille du registre.
nnnn ..... Nombre maximal de registres.
tt ..... Type de fichier (1-Linéaire, 2-Cyclique).
cc ..... Nombre de champs par registre.
dd ..... Offset tableau des champs.
xx ..... Non utilisé.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : Cette commande permet d'obtenir les données les plus générales à partir desquelles sont définis les fichiers. Si "tt" = "0xff", cela signifie que le fichier dont la structure a été demandée est un fichier non défini dans le programme de l'automate.

☑ **Exemple :** Demande concernant le format du fichier 0 de l'unité 1.

MESSAGE PC -> MIDA : **/01200021;**

01 Numéro de périphérique (1).
20 Code du message.
00 Demande concernant le format du fichier 0.
21 Check-sum du message.

MESSAGE MIDA -> PC : **/01A00000001500010108 0AFF00BF;**

01 Numéro de périphérique (1).
A0 Réponse au message 20.
0000 Offset fichier (0 octet).
0015 Taille du registre (21 octets).
0001 Nombre maximal de registres (1).
01 Fichier linéaire.
08 Nombre de champs par registre (8 champs).
0A Offset tableau des champs (10).
FF Non utilisé.
00 Octet d'état (absence d'erreur).
BF Check-sum du message.

MESSAGE 21 :
DEMANDE CONCERNANT LES CHAMPS D'UN FICHIER

Message 21	
Question	Réponse
/NP 21 nn cc CK;	Rép. OK /NP A1 rrrr tt xx ... rrrr tt xx ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
nn ..... Offset tableau des champs
cc ..... Nombre de champs.
rrrr ..... Registre associé ou taille s'il s'agit d'un champ de type Texte.
tt ..... Type de champ (0-Relais, 1-Entier, 2-À virgule flottante, 3-Texte, 4-Date, 5-Heure).
xx ..... Non utilisé.
CK ..... Check-sum du message.

REMARQUE : Les champs de tous les fichiers sont définis dans une table de champs. Le premier champ d'un fichier donné est défini par la donnée "Offset tableau des champs" du Message 20 (Demande concernant le format d'un fichier). Une demande peut se référer à un nombre maximal de 15 champs.

☑ **Exemple :** Demande concernant le format des champs du fichier 0 de l'unité 1.

MESSAGE PC -> MIDA : **/012100082A;**

01 Numéro de périphérique (1).
21 Code du message.
00 Offset tableau des champs (00).
08 Nombre de champs (8).
2A Check-sum du message.

MESSAGE MIDA -> PC :

/01A1019000FF019100FF000002FF012C01FF012D01FF002A05FF0 00503FF002C04FF0083;

01 Numéro de périphérique (1).
A1 Réponse au message 21.
0190 Registre associé (400).
00 Type de champ (Relais).
FF Non utilisé.
0005 Registre associé (chaîne de 5 caractères)
03 Type de champ (Texte).
FF Non utilisé.
0191 Registre associé (401).
00 Type de champ (Relais).
FF Non utilisé.
0000 Registre associé (0).
02 Type de champ (Registre à virgule flottante).
FF Non utilisé.
012C Registre associé (300).
01 Type de champ (Registre entier).
FF Non utilisé.
012D Registre associé (301).
01 Type de champ (Registre entier).
FF Non utilisé.
002A Registre associé (42).
05 Type de champ (Heure).
FF Non utilisé.
0028 Registre associé (40).
04 Type de champ (Date).
FF Non utilisé.
00 Octet d'état (absence d'erreur).
83 Check-sum du message.

MESSAGE 22 :
DEMANDE CONCERNANT UN BLOC MÉMOIRE DE FICHIERS

Message 22	
Question	Réponse
/NP 22 pppp nn CK;	Rép. OK /NP A2 NN ... NN ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
pppp ..... Offset fichier.
nn ..... Demande concernant le nombre d'octets.
NN ..... Octet lu en HEXASCII.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : Lit directement le contenu de la mémoire des fichiers. Le premier octet d'un fichier donné se trouve sur la position définie par la donnée "Offset fichier" du Message 20 (Demande concernant le format d'un fichier). Si vous commencez à numéroter les registres du fichier à partir de 0, vous devrez donc calculer son offset et le nombre d'octets pour lire un registre donné.

Offset = Registre * Taille_du_registre + Offset_fichier

Nombre_d'_octets = Taille_du_registre

Tenez compte du fait que la taille des blocs à lire ne doit pas être supérieure à 60 octets.

Pour déterminer le format dans lequel le contenu de la mémoire est reçu, consultez la section "Format des données dans la mémoire des fichiers", qui se trouve à la fin de ce chapitre.

☑ **Exemple :** Demande concernant un bloc mémoire de fichiers de l'unité 1.

MESSAGE PC -> MIDA : **/01220000193C;**

01 Numéro de périphérique (1).
22 Code du message.
0000 Offset fichier (0).
19 Demande concernant le nombre d'octets (25).
3C Check-sum du message.

MESSAGE MIDA -> PC :

/01A200202020202000CDCCF6426F00DE00252E0E1D0363000000000045;

01	Numéro de périphérique (1).
A2	Réponse au message 22.
00	Relais 400 (0).
20 20 20 20 20	Texte ( ).
00	Relais 401 (0).
CD CC F6 42	Registre à virgule flottante 0 (123.400).
6F 00	Registre entier 300 (111).
DE 00	Registre entier 301 (222).
25 2E 0E	Registre Heure (37 s, 46 min, 14h00).
1D 03 63	Registre Date (29, 03, 99)
00	Vide.
00	Vide.
00	Vide.
00	Vide.
00	Octet d'état (absence d'erreur).
45	Check-sum du message.

MESSAGE A2 :
ENREGISTREMENT D'UN BLOC MÉMOIRE DE FICHIERS

Message A2	
Question	Réponse
/NP A2 pppp dd ... dd CK;	Rép. OK /NP 00 ST CK; Rép. NOK /NP FF ST CK;

où :

NP ..... Numéro de périphérique.
pppp ..... Offset fichier.
dd ..... Octet à enregistrer en mémoire.
ST ..... État de l'équipement.
CK ..... Check-sum du message.

REMARQUE : Enregistre directement des données dans la mémoire des fichiers. Le premier octet d'un fichier donné se trouve sur la position définie par la donnée "Offset fichier" du Message 20 (Demande concernant le format d'un fichier). Si vous commencez à numéroter les registres du fichier à partir de 0, vous devrez donc calculer son offset et le nombre d'octets pour lire un registre donné.

Offset = Registre * Taille_du_registre + Offset_fichier

Nombre_d'_octets = Taille_du_registre

Tenez compte du fait que la taille des blocs à lire ne doit pas être supérieure à 60 octets.

Pour déterminer le format dans lequel le contenu de la mémoire est reçu, consultez la section "Format des données dans la mémoire des fichiers", qui se trouve à la fin de ce chapitre.

☑ **Exemple :** Enregistrement d'un bloc mémoire de fichiers de l'unité 1.

MESSAGE PC -> MIDA :

/01A20000004152524554FFCDCCF642D2042E160000121D0363A0;

01	Numéro de périphérique (1).
A2	Code du message.
0000	Offset fichier.
00	Relais 400 (0).
41 52 52 45 54	Texte (ARRÊT).
FF	Relais 401 (1).
CD CC F6 42	Registre à virgule flottante 0 (123.400).
D2 04	Registre entier 300 (1234).
2E 16	Registre entier 301 (5678).
00 00 12	Registre Heure (00 s, 00 min, 18h00).
1D 03 63	Registre Date (29, 03, 99)
A0	Check-sum du message.

MESSAGE MIDA -> PC : **/01000001;**

01	Numéro de périphérique (1).
00	Confirmation de la modification OK.
00	Octet d'état (absence d'erreur).
01	Check-sum du message.

FORMAT DES DONNÉES DANS LA MÉMOIRE DES FICHIERS

TYPE RELAIS

Taille : 1 octet
Encapsulage : FF - relais ON
00 - relais OFF

TYPE REGISTRE ENTIER

Taille : 2 octets
Encapsulage : Valeur du registre entier (directement).

TYPE REGISTRE À VIRGULE FLOTTANTE

Taille : 4 octets
Encapsulage : Valeur du registre à virgule flottante au format IEEE.

TYPE TEXTE

Taille : Variable selon la définition de ce champ.
Encapsulage : 1 octet par caractère de texte.

TYPE HEURE

Taille : 3 octets
Encapsulage :

Octet 0	Bits 0..4 Bits 5..7	Date Jour de la semaine
Octet 1	Bits 0..7	Mois
Octet 2	Bits 0..7	Année (00..99)

TYPE DATE

Taille : 3 octets
Encapsulage :

Octet 0	Bits 0..4	Secondes
Octet 1	Bits 0..5	Minutes
Octet 2	Bits 0..5	Heure

✓ PROTOCOLE MODBUS EN MODE RTU

En mode RTU, les messages commencent par un silence dont la durée est au moins égale au temps nécessaire pour envoyer 3,5 caractères et dépend de la vitesse de transmission (en bauds).

Les quatre parties de chaque message - numéro de périphérique, commande, données et CRC - sont transmises après de silence.

Le message se termine par un autre silence de 3,5 caractères.

Début	N° périphérique	Commande	Donnée	CRC	Fin
Silence	8 bits	8 bits	n × 8 bits	16 bits	Silence

✓ NUMÉRO DE PÉRIPHÉRIQUE

Le protocole MODbus admet des adresses valides de 0 à 247 (décimal). L'adresse 0 est réservée à la transmission simultanée de commande à tous les périphériques (broadcasting). Les équipements MIDA admettent uniquement des adresses valides de 0 à 99.

✓ COMMANDE

Les commandes valides vont de 1 à 255 (décimal). Les équipements MIDA disposent des commandes suivantes :

Commande	Description
Commandes 01 et 02	Lecture de relais de forme compact
Commandes 03 et 04	Lecture de registres entiers
Commande 05	Écriture d'un relais
Commande 06	Écriture d'un registre entier
Commande 0F	Écriture de plusieurs relais de forme compact
Commande 10	Écriture de plusieurs registres entiers

COMMANDES 01 et 02 :

LECTURE DE RELAIS DE FORME COMPACT

FORMAT :

Question	Exemple (Hex)*
Nº périphérique	63
Commande	01
Adresse début (Hi)	00
Adresse début (Lo)	FA
Nº Relais (Hi)	00
Nº Relais (Lo)	10
CRC (Lo)	15
CRC (Hi)	B5

(*) Demander 16 relais (à partir du 250) au périphérique 99.

Réponse	Exemple (Hex)
Nº périphérique	63
Commande	01
Nbre d'octets de données	02
Donnée 0 (relais 257-250)	6B
Donnée 1 (relais 265-258)	5C
CRC (Lo)	6F
CRC (Hi)	3D

COMMANDES 03 et 04 :

LECTURE DE REGISTRES ENTIERS

FORMAT :

Question	Exemple (Hex)*
Nº périphérique	63
Commande	03
Adresse début (Hi)	00
Adresse début (Lo)	FA
Nº Registres (Hi)	00
Nº Registres (Lo)	02
CRC (Lo)	EC
CRC (Hi)	78

(*) Demander 2 registres entiers (à partir du 250) au périphérique 99.

Réponse	Exemple (Hex)*
Nº périphérique	63
Commande	03
Nbre d'octets de données	04
Donnée (Hi) (Reg. 250)	6B
Donnée (Lo) (Reg. 250)	5C
Donnée (Hi) (Reg. 251)	00
Donnée (Lo) (Reg. 251)	01
CRC (Lo)	A4
CRC (Hi)	03

COMMANDE 05 :

ÉCRITURE D'UN RELAIS :

FF00 – Relais ON
0000 – Relais OFF

FORMAT :

Question	Exemple (Hex)*
Nº périphérique	63
Commande	05
Adresse début (Hi)	00
Adresse début (Lo)	FA
Valeur du relais (Hi)	FF
Valeur du relais (Lo)	00
CRC (Lo)	A4
CRC (Hi)	49

(*) Activer le relais 250 (ON) du périphérique 99.

Réponse	Exemple (Hex)
Nº périphérique	63
Commande	05
Adresse début (Hi)	00
Adresse début (Lo)	FA
Valeur du relais (Hi)	FF
Valeur du relais (Lo)	00
CRC (Lo)	A4
CRC (Hi)	49

COMMANDE 06 :

ÉCRITURE D'UN REGISTRE ENTIER

FORMAT :

Question	Exemple (Hex)*
Nº périphérique	63
Commande	06
Adresse début (Hi)	00
Adresse début (Lo)	FA
Valeur du registre (Hi)	03
Valeur du registre (Lo)	E8
CRC (Lo)	A1
CRC (Hi)	07

(*) Écrire le registre 250 (1000) du périphérique 99.

Réponse	Exemple (Hex)
Nº périphérique	63
Commande	06
Adresse début (Hi)	00
Adresse début (Lo)	FA
Valeur du registre (Hi)	03
Valeur du registre (Lo)	E8
CRC (Lo)	A1
CRC (Hi)	07

COMMANDE 0F :

ÉCRITURE DE PLUSIEURS RELAIS DE FORMES COMPACT

FORMAT :

Question	Exemple (Hex)*
Nº périphérique	63
Commande	0F
Adresse début (Hi)	00
Adresse début (Lo)	FA
Nº Relais (Hi)	00
Nº Relais (Lo)	10
Nbre d'octets de données	02
Donnée 0 (relais 257-250)	6B
Donnée 1 (relais 265-258)	5C
CRC (Lo)	68
CRC (Hi)	11

(*) Forcer 16 relais (à partir du 250) du périphérique 99.

Réponse	Exemple (Hex)
Nº périphérique	63
Commande	0F
Adresse début (Hi)	00
Adresse début (Lo)	FA
Nº Relais (Hi)	00
Nº Relais (Lo)	10
CRC (Lo)	7C
CRC (Hi)	74

COMMANDE 10 :

ÉCRITURE DE PLUSIEURS REGISTRES ENTIERS

FORMAT :

Question	Exemple (Hex)*
Nº périphérique	63
Commande	10
Adresse début (Hi)	00
Adresse début (Lo)	FA
Nº Registres (Hi)	00
Nº Registres (Lo)	02
Nbre d'octets de données	04
Donnée (Hi) (Registre 250)	6B
Donnée (Lo) (Registre 250)	5C
Donnée (Hi) (Registre 251)	00
Donnée (Lo) (Registre 251)	01
CRC (Lo)	94
CRC (Hi)	DB

(*) Forcer 2 registres entiers (à partir du 250) du périphérique 99.

Réponse	Exemple (Hex)
Nº périphérique	63
Commande	10
Adresse début (Hi)	00
Adresse début (Lo)	FA
Nº Registres (Hi)	00
Nº Registres (Lo)	02
CRC (Lo)	69
CRC (Hi)	BB

✓ CRC

Le CRC (Contrôle de redondance cyclique) permet de vérifier si le message est correctement arrivé à destination. Il convient de l'insérer dans tous les messages reçus. Lorsqu'un équipement reçoit un message avec un CRC incorrect, il ne répond pas et la question doit être répétée au bout d'un certain temps. Il en est de même si la réponse reçue ne contient pas le CRC approprié.

Vous trouverez, ci-après, deux routines (langage C) de calcul du CRC. La première est courte et assez lente, et la deuxième est plus rapide mais aussi plus longue.

ROUTINE 1

```
unsigned int crc(unsigned char *dat,int nudat)
{
int n;
unsigned int CRC=0xffff;
unsigned int CRCO;

while(nudat--)
        {
    CRC=CRC^*dat++;
    for (n=0;n<8;n++)
            {
        CRCO=CRC;
        CRC>>=1;
        if (CRCO&1) CRC^=0xa001;
        }
    }
return(CRC);
}
```

ROUTINE 2

```
unsigned char thi[256]={
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,
0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,
0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,
0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,
0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,
0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40};



unsigned char tlo[256]={
0x00,0xc0,0xc1,0x01,0xc3,0x03,0x02,0xc2,0xc6,0x06,0x07,0xc7,0x05,0xc5,0xc4,0x04,
0xcc,0x0c,0x0d,0xcd,0x0f,0xcf,0xce,0x0e,0x0a,0xca,0xcb,0x0b,0xc9,0x09,0x08,0xc8,
0xd8,0x18,0x19,0xd9,0x1b,0xdb,0xda,0x1a,0x1e,0xde,0xdf,0x1f,0xdd,0x1d,0x1c,0xdc,
0x14,0xd4,0xd5,0x15,0xd7,0x17,0x16,0xd6,0xd2,0x12,0x13,0xd3,0x11,0xd1,0xd0,0x10,
0xf0,0x30,0x31,0xf1,0x33,0xf3,0xf2,0x32,0x36,0xf6,0xf7,0x37,0xf5,0x35,0x34,0xf4,
0x3c,0xfc,0xfd,0x3d,0xff,0x3f,0x3e,0xfe,0xfa,0x3a,0x3b,0xfb,0x39,0xf9,0xf8,0x38,
0x28,0xe8,0xe9,0x29,0xeb,0x2b,0x2a,0xea,0xee,0x2e,0x2f,0xef,0x2d,0xed,0xec,0x2c,
0xe4,0x24,0x25,0xe5,0x27,0xe7,0xe6,0x26,0x22,0xe2,0xe3,0x23,0xe1,0x21,0x20,0xe0,
0xa0,0x60,0x61,0xa1,0x63,0xa3,0xa2,0x62,0x66,0xa6,0xa7,0x67,0xa5,0x65,0x64,0xa4,
0x6c,0xac,0xad,0x6d,0xaf,0x6f,0x6e,0xae,0xaa,0x6a,0x6b,0xab,0x69,0xa9,0xa8,0x68,
0x78,0xb8,0xb9,0x79,0xbb,0x7b,0x7a,0xba,0xbe,0x7e,0x7f,0xbf,0x7d,0xbd,0xbc,0x7c,
0xb4,0x74,0x75,0xb5,0x77,0xb7,0xb6,0x76,0x72,0xb2,0xb3,0x73,0xb1,0x71,0x70,0xb0,
0x50,0x90,0x91,0x51,0x93,0x53,0x52,0x92,0x96,0x56,0x57,0x97,0x55,0x95,0x94,0x54,
0x9c,0x5c,0x5d,0x9d,0x5f,0x9f,0x9e,0x5e,0x5a,0x9a,0x9b,0x5b,0x99,0x59,0x58,0x98,
0x88,0x48,0x49,0x89,0x4b,0x8b,0x8a,0x4a,0x4e,0x8e,0x8f,0x4f,0x8d,0x4d,0x4c,0x8c,
0x44,0x84,0x85,0x45,0x87,0x47,0x46,0x86,0x82,0x42,0x43,0x83,0x41,0x81,0x80,0x40}
;
```

```
unsigned int crc(unsigned char *dat,int nudat)
{
unsigned char CRClo=0xff;
unsigned char CRChi=0xff;
unsigned int ind;

while(nudat--)
        {
        ind=CRClo^*dat++;
        CRClo=CRChi^thi[ind];
        CRChi=tlo[ind];
        }
return(((unsigned int)CRChi<<8)|CRClo);
}
```

Certains modèles d'équipements MIDA disposent d'instructions, de registres et de relais destinés à la réception et à la transmission de chaînes ASCII. Le protocole correspondant est appelé "Protocole libre".

Le Protocole libre étant un protocole point à point, il ne fonctionne pas sur les réseaux 485, mais uniquement sur le port de communication RS232 des équipements MIDA.

✓ Configuration del Set-up

Les équipements qui disposent du Protocole libre possèdent quatre paramètres permettant de le configurer (les deux premiers varient selon le modèle d'équipement MIDA) :

- **Protocole COM :** Permet de spécifier le protocole par défaut du MIDA lors de chacune des initialisations de l'équipement (uniquement valable pour les équipements qui disposent d'un port de communication RS232 non configurable en tant que RS485).
- **Protocole libre :** Permet de spécifier quel port de communication est configuré en tant que RS232 et d'activer le Protocole libre (uniquement valable pour les équipements qui disposent de ports de communication RS232 configurables en tant que RS485).
- **Caractère initial :** Permet de spécifier le caractère initial du message.
- **Caractère final :** Permet de spécifier le caractère final du message.
- **Longueur du message :** Permet de spécifier la longueur du message de réception.

REMARQUE :

Si la valeur du paramètre "Caractère final"/"Caractère initial" est "0", le message ne contient pas de caractère final/initial.

Si la valeur du paramètre "Longueur du message" est "0", celle-ci n'est pas vérifiée. Si la valeur introduite est supérieure à 132 (taille maximale de la mémoire-tampon), les derniers caractères sont perdus.

✓ Relais et registres

Les équipements MIDA disposant d'un Protocole libre possèdent un relais et un registre entier destinés à la gestion de ce protocole (consultez la topographie de la mémoire dans le Manuel Utilisateur de l'équipement) :

- **Message complet (relais) :** Ce relais s'active lorsque la mémoire-tampon de réception contient un message à traiter.
- **Activation du Protocole libre (relais) :** Ce relais permet d'activer et de désactiver le Protocole libre.
- **Longueur du message de réception (registre entier) :** Indique le nombre de caractères actuellement stockés dans la mémoire-tampon de réception.

✓ Fonctionnement

Si le port de communication RS232 est en mode Protocole libre (le protocole MIDAbus est alors désactivé), celui-ci fonctionne de la manière suivante :

- Si le caractère initial est différent de zéro, lorsque le système le reçoit, il active la mémoire-tampon de réception et le stocke. Le relais RX s'active et le registre entier Longueur du message de réception passe à "1".
- Si le caractère initial est zéro, le premier caractère reçu après traitement du message initialise la mémoire-tampon de réception et est stocké. Le relais RX s'active et le registre entier Longueur du message de réception passe à "1".
- Si le caractère final est différent de zéro, lorsque le système le reçoit, il active le relais Message complet et désactive la réception jusqu'à ce que le message soit traité.
- Lorsque le message atteint la longueur programmée lors de la configuration, le relais Message complet s'active et la réception se désactive jusqu'à ce que le message soit traité.
- Il peut arriver que le relais Message complet s'active sans que le relais RX soit activé lorsque le message possède un caractère initial et une longueur programmée. Cela implique que le système a reçu un message de la longueur programmée sans avoir

préalablement reçu le caractère initial. Le cas échéant, c'est le programme utilisateur qui doit agir en conséquence.

Mis à part l'ajout de CARRY RETURN (13 déc.) à la fin du message, la routine de transmission fonctionne de la même manière en mode Protocole libre et sans ce protocole.

REMARQUE : Chaque port de communication possède une seule mémoire-tampon qui assure à la fois la transmission et la réception. Cela signifie donc que si vous tentez de transmettre un message via un port de communication donné alors que celui-ci est en train d'en recevoir un autre, vous perdrez la partie reçue. Nous vous conseillons donc de vous assurer que le relais RX n'est pas activé (c'est-à-dire qu'aucune réception n'est en cours) avant de lancer toute transmission pour éviter de perdre des messages.

✓ Activation et désactivation du Protocole libre

Le relais Activation du Protocole libre permet d'activer et de désactiver celui-ci par l'intermédiaire du programme. Vous pouvez donc travailler en mode Protocole libre et passer en fonctionnement normal (MIDAbus) en un point du programme.

Le Protocole libre étant activé (relais activé), si vous désactivez le relais, vous désactivez également le Protocole libre. Si vous réactivez le relais, vous réactivez le Protocole libre.

Lorsque vous réactivez le Protocole libre, nous vous conseillons de vérifier le contenu de la mémoire-tampon et d'exécuter immédiatement l'instruction "COM9" pour détecter la présence éventuelle de données parasites et réinitialiser la mémoire-tampon de réception.

Dans le cas des équipements MIDA dont les ports de communication RS232 peuvent être configurés en tant que RS485, le relais Activation du Protocole libre ne fonctionnera que si ce protocole a été activé dans la configuration de l'équipement. Il n'en est pas de même pour les équipements MIDA dont les ports de communication RS232 ne peuvent pas être configurés en tant que RS485, car le relais Activation du Protocole libre fonctionne indépendamment de la configuration de l'équipement ; il peut être activé dans la configuration de l'équipement ou dans le programme utilisateur.

DISB

DISB XXXX YYYY

MNÉMONIQUE : **DISB**
OPÉRANDE XXXX : **Définit la base dans laquelle le nombre de caractères est écrit**
OPÉRANDE YYYY : **Nombre de caractères à écrire**
CODE INSTRUCTION : **82**

DESCRIPTION :

Écrit, dans la mémoire-tampon générale et à partir de la position désignée par l'instruction LOC, un nombre stocké dans la pile arithmétique.
L'opérande XXXX indique le type de base dans laquelle l'écriture a lieu dans la mémoire-tampon :

0 – Binaire.
1 – Octal (0..177777).
2 – Décimal. (0..65536).
3 – Hexadécimal (0..FFFF).
4 – Décimal codé binaire (BCD).

L'opérande YYYY indique le nombre de caractères à écrire (de 1 à 6).

Ne fait pas intervenir les piles (0).

☑ Exemples:

Pile arithmétique	Instruction	Mémoire-tampon
65	DISB 0 1	'A'
65	DISB 1 4	'0101'
65	DISB 2 4	'0065'
65	DISB 3 4	'0041'
65	DISB 4 1	'e' Caractère 65_H
16706	DISB 0 1	'B' Caractère 42_H
16706	DISB 0 2	'AB' Caractères 41_H42_H
16706	DISB 1 6	'040502'

16706	DISB 1 5	‘40502’
16706	DISB 1 4	‘0502’
16706	DISB 2 6	‘016706’
16706	DISB 2 5	‘16706’
16706	DISB 2 4	‘6706’
16706	DISB 3 6	‘004142’
16706	DISB 3 5	‘04142’
16706	DISB 3 4	‘4142’
16706	DISB 3 3	‘142’
16706	DISB 4 3	Caractères $01_H67_H06_H$
16706	DISB 4 2	Caractères 67_H06_H
16706	DISB 4 1	Caractère 06_H

LECB

LECB XXXX YYYY

MNÉMONIQUE : **LECB**
OPÉRANDE XXXX : **Définit la base dans laquelle le nombre de caractères est lu**
OPÉRANDE YYYY : **Nombre de caractères à lire**
CODE INSTRUCTION : **83**

DESCRIPTION :

Lit un nombre dans la mémoire-tampon générale, à partir de la position désignée par l'instruction LOC, et le stocke dans la pile arithmétique.
L'opérande XXXX indique le type de base dans laquelle la lecture a lieu dans la mémoire-tampon :

0 – Binaire.
1 – Octal (0..177777).
2 – Décimal. (0..65536).
3 – Hexadécimal (0..FFFF).
4 – Décimal codé binaire (BCD).

L'opérande YYYY indique le nombre de caractères à lire (de 1 à 6). Ne fait pas intervenir les piles (0).

☑ Exemple :

Mémoire-tampon	**Instruction**	**Pile arithmétique**
'A'	LECB 0 1	65
'0101'	LECB 1 4	65 = 101_{OCT}
'0065'	LECB 2 4	65
'0041'	LECB 3 4	65 = 41_{H}
'e' Caractère 65_{H}	LECB 4 1	65
'AB' Caractères $41_{H}42_{H}$	LECB 0 1	65 = 41_{H}
'AB' Caractères $41_{H}42_{H}$	LECB 0 2	16706 = 4142_{H}
'040502'	LECB 1 6	16706 = 040502_{OCT}
'040502'	LECB 1 5	2088 = 04050_{OCT}

'040502'	LECB 1 4	261 = 0405_{OCT}
'016706'	LECB 2 6	016706
'016706'	LECB 2 5	01670
'016706'	LECB 2 4	0167
'4142'	LECB 3 4	16706 = 4142_H
'4142'	LECB 3 3	1044 = 414_H
'4142'	LECB 3 2	65 = 41_H
Caractères $01_H67_H06_H$	LECB 4 3	016706
Caractères $01_H67_H06_H$	LECB 4 2	0167
Caractères $01_H67_H06_H$	LECB 4 1	01

Le programme de l'automate doit veiller à ce que le nombre lu puisse être stocké dans un registre entier (16 bits). Dans le cas contraire, le résultat est imprévisible.

Le nombre stocké dans la pile arithmétique sera traité comme un entier avec signe, ce qui signifie que si le nombre écrit dans la mémoire-tampon est supérieur à 32767, lorsqu'il sera stocké dans la pile, vous obtiendrez le nombre négatif correspondant. Exemple : si la mémoire-tampon contient un 65535 en décimal ou un FFFF en hexadécimal, c'est un -1 qui sera stocké dans la pile.

COM

COM XXXX

MNÉMONIQUE : **COM**
OPÉRANDE XXXX : **Constante 9**
CODE INSTRUCTION : **80**

DESCRIPTION :

Copie le contenu actuel de la mémoire-tampon de réception dans la mémoire-tampon générale pour qu'il puisse être lu à l'aide de LECB et écrit à l'aide de DISB.
L'opérande XXXX est la constante 9 : indique au port de communication qu'il s'agit d'une opération utilisant les routines du Protocole libre.
Ne fait pas intervenir les piles (0).

Les différentes phases d'exécution de l'instruction COM 9 sont :

- Copie du contenu de la mémoire-tampon de réception sur la mémoire-tampon générale.
- Désactivation du relais RX.
- Désactivation du relais Message complet.
- Mise à zéro du registre entier Longueur du Message.
- Exécution de l'instruction LOC 0.

Supposons que le relais RX soit activé, ce qui indique qu'un processus de réception est en cours et que vous perdrez le contenu de la mémoire-tampon de réception si vous transmettez un message.

Le relais Message complet indique que le message est prêt à être transféré de la mémoire-tampon de réception à la mémoire-tampon générale (exécution de l'instruction COM 9). Cela empêche la réception de nouveaux messages pour éviter de perdre celui que vous avez reçu. Après exécution de l'instruction COM 9, ce relais se désactive et le port de communication peut recevoir de nouveaux messages. Vous pouvez également activer et désactiver ce port par l'intermédiaire d'un SET ou d'un RESET du relais.

L'exécution de l'instruction COM 9 entraîne la mise à 0 du registre entier Longueur du message de RX. S'il s'avère nécessaire d'utiliser cette valeur pour le traitement du message, il convient donc de l'enregistrer dans un registre quelconque avant d'utiliser cette instruction.

TE

✓ ERREURS DE DURÉE D'EXÉCUTION

Les erreurs de durée d'exécution sont celles qui se produisent pendant l'exécution du programme utilisateur du MIDA. Celle-ci débute lorsque sa compilation et son transfert à l'équipement ont eu lieu sans erreur. Ce chapitre contient une description des erreurs détectées pendant l'exécution du programme.

Un programme correctement écrit du point de vue de la syntaxe et dont la compilation n'occasionne aucune erreur peut néanmoins ne pas fonctionner correctement après son transfert au MIDA ou être à l'origine d'erreurs et de dysfonctionnements de l'équipement (réinitialisation automatique, arrêts, etc.). Le cas échéant, procédez comme suit pour tester chacune des possibilités décrites dans ce paragraphe.

Il existe divers types d'erreurs :

TYPE D'ERREUR	ÉLÉMENT DÉTECTEUR
Logicielle (Software)	Relais d'erreurs.
Matérielle (Hardware)	Registres entiers d'erreurs.
Communication	Octet d'état du protocole MIDAbus
État des ports de communication	Relais d'état des ports de communication.

REMARQUE : Les relais/registres d'erreurs ou d'état dont disposent les équipements MIDA varient selon le modèle. Pour déterminer ceux dont est muni votre équipement, consultez la topographie de la mémoire dans le Manuel Utilisateur de celui-ci.

✓ ERREURS LOGICIELLES (SOFTWARE)

Ces erreurs se produisent lorsque le programme utilisateur du MIDA est mal écrit, mal structuré, etc.

Chaque fois qu'il exécute l'instruction END, l'équipement initialise les niveaux des piles logiques, arithmétique et de sous-routines, et actualise les relais d'erreurs pour empêcher toute interruption de l'exécution du programme.

L'équipement ne s'arrête donc pas et exécute la partie conforme du programme, mais ne garantit en aucun cas le fonctionnement correct de l'installation.

Une série de relais internes indiquent à tout moment les erreurs détectées lors de l'exécution de l'instruction END.

C'est l'exécution de l'instruction END qui entraîne l'activation des relais internes d'erreurs. Ceux-ci ne sont donc jamais activés si cette instruction n'est pas exécutée.

Un relais d'erreur activé ne peut être désactivé que par l'intervention de l'utilisateur sur le relais correspondant ou par la réinitialisation ou la mise sous tension de l'équipement. Cela signifie que si une erreur disparaît, le relais correspondant ne se désactivera pas avant que l'une des opérations ci-dessus soit effectuée.

Les relais d'erreurs de l'équipement sont les suivants :

Erreur de relais de batterie

Ce relais indique les erreurs relatives à la batterie au Ni-Cd qui préserve les données contenues dans la RAM et celles de l'horloge interne de l'équipement.
Ce relais fonctionne comme tout autre relais interne.

État du relais	Signification
1	Batterie en bon état.
0	Erreur de batterie (panne ou batterie insuffisamment chargée).

L'erreur détectée au niveau de la batterie est due à un dysfonctionnement matériel : la charge de la batterie a atteint la limite minimale permise ou la batterie est en mauvais état.

Pour vérifier l'état de la batterie, mettez l'horloge interne à l'heure et déconnectez l'équipement MIDA pendant quelques minutes. Rebranchez ensuite l'équipement : si l'heure indiquée est incorrecte, la batterie est défectueuse.

Erreur de division par zéro

Cette erreur se produit lorsque le programme MIDA tente d'effectuer une division par zéro.
Ce relais fonctionne comme tout autre relais interne.

État du relais	Signification
0	Absence d'erreur.
1	Tentative de division par zéro.

Vérifiez le programme pour déterminer l'endroit où a lieu cette tentative de division par zéro et remédiez à cette erreur.
Le résultat d'une tentative de division par zéro peut s'avérer imprévisible.

Erreur de dépassement arithmétique (overflow)

Cette erreur se produit lorsque le programme MIDA tente d'effectuer une opération arithmétique dont le résultat implique un dépassement de la capacité de calcul de l'équipement.
Ce relais fonctionne comme tout autre relais interne.

État du relais	Signification
0	Absence d'erreur.
1	Tentative d'exécution d'une opération arithmétique dont le résultat implique un dépassement de la capacité de calcul de l'équipement.

Vérifiez le programme pour déterminer l'endroit où a lieu ce calcul arithmétique et remédiez à cette erreur.

Le résultat d'une opération arithmétique de ce type peut s'avérer imprévisible.

Erreur de pile logique

Cette erreur se produit lorsqu'il existe un déséquilibre entre les chargements et les déchargements effectués sur la pile logique de l'équipement.
Ce relais fonctionne comme tout autre relais interne.

État du relais	Signification
0	Absence d'erreur.
1	Existence d'un déséquilibre entre les chargements et les déchargements effectués sur la pile logique de l'équipement.

Vérifiez le programme pour déterminer l'endroit où a lieu ce déséquilibre.

Erreur de pile arithmétique

Cette erreur se produit lorsqu'il existe un déséquilibre entre les chargements et les déchargements effectués sur la pile arithmétique de l'équipement.
Ce relais fonctionne comme tout autre relais interne.

État du relais	Signification
0	Absence d'erreur.
1	Existence d'un déséquilibre entre les chargements et les déchargements effectués sur la pile arithmétique de l'équipement.

Vérifiez le programme pour déterminer l'endroit où a lieu ce déséquilibre.

Erreur de pile de sous-routines

Cette erreur se produit lorsqu'il existe un déséquilibre entre les appels de sous-routines et les retours de celles-ci, c'est-à-dire lorsque le programme contient un appel à une sous-routine et que celle-ci ne se termine pas par l'instruction de retour (RET).
Ce relais fonctionne comme tout autre relais interne.

État du relais	Signification
0	Absence d'erreur.
1	Existence d'un déséquilibre entre les appels de sous-routines et les retours de celles-ci.

Vérifiez le programme pour déterminer l'endroit où a lieu ce déséquilibre.

✓ ERREURS MATÉRIELLES (HARDWARE)

La mémoire de l'équipement comporte des registres de détection d'erreurs qui indiquent tous les incidents susceptibles de se produire lors de la mise en marche de l'équipement et lors de l'exécution des programmes MIDA.

En principe, les erreurs indiquées dans ces registres sont celles susceptibles de se produire au niveau du Hardware de l'équipement de base et des modules ou des cartes d'extension (modules mal insérés ou paramètres non conformes, mémoires EPROM ou EEPROM en mauvais état, etc.), mais il peut également s'agir d'erreurs provoquées par l'exécution du programme MIDA. Ces informations font alors double emploi avec celles fournies par les relais de détection d'erreurs décrits précédemment.

Nous détaillons plus avant la manière dont le contenu de ces registres indique les erreurs détectées.

Ces registres sont accessibles depuis le programme MIDA et depuis tout PC connecté via un port série. Les routines de traitement des erreurs peuvent donc être exécutées depuis l'équipement lui-même ou depuis un PC central qui communique les problèmes détectés à l'utilisateur.

L'équipement signale les erreurs détectées en activant les bits de ces registres. Chaque bit indique un type d'erreur différent et chaque registre peut détecter plus d'une erreur.

Le contenu du registre indique les erreurs détectées.

C'est l'exécution de l'instruction END qui entraîne l'activation des bits des registres d'erreurs. Ceux-ci ne sont donc jamais actualisés si cette instruction n'est pas exécutée.

Un bit de registre activé ne peut être désactivé que par l'intervention de l'utilisateur ou par la mise sous tension de l'équipement. Cela signifie que si une erreur disparaît, le registre correspondant ne sera pas actualisé pas avant que l'une des opérations ci-dessus ait été effectuée.

Les seuls bits qui seront automatiquement actualisés seront le bit B1 d'erreur générale et les bits d'erreurs de cartes des équipements qui en disposent.

Vous trouverez, ci-après, une description du contenu de ces registres ainsi que des erreurs détectées :

Registre	Description
Erreurs générales	Registre général de détection des erreurs. Elles peuvent être dues à un dysfonctionnement de la carte du microprocesseur ou des modules ou cartes d'extension ou à une erreur dans le programme utilisateur du MIDA. Son code permet de déterminer la nature de l'erreur.
Erreurs de la carte-mère	Registre de détection des erreurs de la carte-mère de l'équipement MIDA. Ce registre contient toutes les erreurs susceptibles d'être détectées au niveau de la carte-mère.
Erreurs des cartes	Registre de détection des erreurs des modules ou des cartes d'extension.

REMARQUE : Les bits d'erreurs fournis par les registres d'erreurs varient selon le modèle d'équipement MIDA. Pour déterminer ceux qui correspondent à votre équipement, consultez le Manuel Utilisateur de celui-ci.

Registre des erreurs générales

Indique les erreurs

Registre de 2 octets (16 bits) (X = bit non utilisé) :

7	6	X	X	X	2	1	0	7	6	5	4	3	2	1	0
OCTET SUPÉRIEUR								**OCTET INFÉRIEUR**							

BIT	Valeur en hexad.	Poids en déc.	Type d'erreur et solution
b0	0001	1	Demande d'exécution d'une opération impossible à exécuter car le programme MIDA est en marche. Cette erreur se produit uniquement lorsque les demandes sont effectuées par l'intermédiaire des protocoles de communication. Supprimez la demande ou interrompez l'exécution du programme MIDA.
b1	0002	2	Erreur au niveau des variables demandées par l'intermédiaire du protocole de communication. Solution : vérifiez les variables demandées dans le message de communication.
b2	0004	4	Tentative de lecture ou d'écriture en dehors de limites de la base de données. Vérifiez la définition des fichiers de la base de données de l'équipement (pseudo-instruction de compilation "FILE").
b3	0008	8	Tentative de lecture ou d'écriture en dehors de limites fixées par la topographie de la mémoire disponible de l'équipement par l'intermédiaire du programme utilisateur. Vérifiez le programme MIDA, et plus particulièrement les instructions d'adressage indirect.
b4	0010	16	Tentative de division par zéro ou opération arithmétique dépassant la capacité de calcul de l'équipement. Vérifiez le programme utilisateur MIDA.

BIT	Valeur en hexad.	Poids en déc.	Type d'erreur et solution
b5	0020	32	Erreur de déséquilibre des piles (logique, arithmétique, de sous-routines). Vérifiez le programme MIDA.
b6	0040	64	Erreur de total de contrôle (Checksum) dans l'EPROM FLASH. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.
b7	0080	128	Erreur de lecture ou d'écriture dans l'EEPROM. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.
B0	0100	256	Erreur dans l'un des microprocesseurs de contrôle d'un module ou d'une carte d'extension. Assurez-vous que l'équipement reconnaît tous les modules et toutes les cartes d'extension insérés (message d'initialisation) pour détecter d'éventuels problèmes d'installation ou de paramétrisation de ceux-ci. Vérifiez également le registre entier des erreurs de cartes pour déterminer le type d'erreur indiqué. Si, malgré ces vérifications, il s'avère impossible de détecter visuellement l'erreur ou si celle-ci persiste, contactez le S.A.T.
B1	0200	512	Erreur d'accès au LCD. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.

BIT	Valeur en hexad.	Poids en déc.	Type d'erreur et solution
B2	0400	1024	Erreur d'accès à l'horloge de l'équipement MIDA. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.
B6	4000	16384	Erreur de communication entre les microprocesseurs des modules ou des cartes d'extension et le microprocesseur de contrôle de ces modules ou de ces cartes. Assurez-vous que l'équipement reconnaît tous les modules et toutes les cartes d'extension insérés (message d'initialisation) pour détecter d'éventuels problèmes d'installation ou de paramétrisation de ceux-ci. Si, malgré ces vérifications, il s'avère impossible de détecter visuellement l'erreur ou si celle-ci persiste, contactez le S.A.T.
B7	8000	32768	Erreur de numérotation répétée de périphérique au niveau d'un module ou d'une carte d'extension. Assurez-vous que l'équipement reconnaît tous les modules et toutes les cartes d'extension insérés (message d'initialisation) pour détecter d'éventuels problèmes d'installation ou de paramétrisation de ceux-ci. Si, malgré ces vérifications, il s'avère impossible de détecter visuellement l'erreur ou si celle-ci persiste, contactez le S.A.T.

** S.A.T. : Service d'Assistance Technique.*

Les bits B3, B4 et B5 ne sont pas utilisés dans ce registre.

En cas de détection simultanée de plusieurs erreurs, le registre prendra la valeur correspondant à la somme des poids décrits dans la colonne POIDS EN DÉCIMAL du tableau.

Registre des erreurs de la carte-mère

Registre de 2 octets (16 bits) (X = bit non utilisé) :

X	X	X	X	X	X	X	X	X	6	X	X	3	2	1	0
OCTET SUPÉRIEUR								**OCTET INFÉRIEUR**							

BIT	Valeur en hexad.	Poids en déc.	Type d'erreur et solution
b0	0001	1	Le PIC-Master n'accepte pas ou ne reconnaît pas l'un des modules ou l'une des cartes d'extension. Assurez-vous que l'équipement reconnaît tous les modules et toutes les cartes d'extension insérés (message d'initialisation) pour détecter d'éventuels problèmes d'installation ou de paramétrisation de ceux-ci. Si, malgré ces vérifications, il s'avère impossible de détecter visuellement l'erreur ou si celle-ci persiste, contactez le S.A.T.
b1	0002	2	RAM de PIC-Master insuffisante pour gérer d'autres modules ou cartes d'extension. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.
b2	0004	4	Erreur de communication entre le PIC-

BIT	Valeur en hexad.	Poids en déc.	Type d'erreur et solution
			Master et le convertisseur A/N de l'un de modules ou de l'une des cartes d'extension. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.
b3	0008	8	Erreur logicielle interne. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.
b6	0040	64	Erreur de numérotation répétée de périphérique au niveau d'un module ou d'une carte d'extension. Assurez-vous que l'équipement reconnaît tous les modules et toutes les cartes d'extension insérés (message d'initialisation) pour détecter d'éventuels problèmes d'installation ou de paramétrisation de ceux-ci. Si, malgré ces vérifications, il s'avère impossible de détecter visuellement l'erreur ou si celle-ci persiste, contactez le S.A.T.

Registre des erreurs de cartes

Registre de 2 octets (16 bits) (X = bit non utilisé) :

X	X	X	X	X	X	X	X	X	X	X	X	3	2	1	0
OCTET SUPÉRIEUR								**OCTET INFÉRIEUR**							

BIT	Valeur en hexad.	Poids en déc.	Type d'erreur et solution
b0	0001	1	Modèle de carte non admis ou configuration de carte non reconnue. Assurez-vous que l'équipement reconnaît tous les modules et toutes les cartes d'extension insérés (message d'initialisation) pour détecter d'éventuels problèmes d'installation ou de paramétrisation de ceux-ci. Si, malgré ces vérifications, il s'avère impossible de détecter visuellement l'erreur ou si celle-ci persiste, contactez le S.A.T.
b1	0002	2	Erreur générale I2C de la carte. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.
b2	0004	4	Erreur de paramétrisation de la carte. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.
b3	0008	8	Erreur interne du convertisseur A/N. Redémarrez l'équipement. Si l'erreur persiste, contactez le S.A.T.

✓ ERREURS DE COMMUNICATION

Les réponses aux messages du protocole MIDAbus incluent un octet d'état de l'équipement (consultez le chapitre Protocole MIDAbus de ce manuel).
Si cet octet est "00", l'équipement ne présente ni problème, ni erreur, mais si l'un des bits est activé, cela traduit un problème de l'équipement ou une erreur de programme.
L'octet d'état correspond aux bits b0 à b7 du tableau des erreurs générales.

✓ RELAIS ÉTAT DES PORTS DE COMMUNICATION

Les relais d'état des ports de communication, Rx COM, Tx COM, CTS COM et TIME-OUT COM reflètent à tout moment l'état dans lequel se trouve chacun des ports de communication de l'équipement.
Intrinsèquement parlant, ces relais ne sont pas des relais d'erreurs, ils reflètent simplement l'état des ports.

Le relais TIME-OUT COM (ports de communication RS232) peut néanmoins être considéré comme un relais d'erreur car il s'active lorsque l'équipement ne peut assurer une communication.
Le TIME-OUT se produit 5 secondes après le début de la tentative de transmission. Ce temps étant écoulé, l'équipement active le relais TIME-OUT COM correspondant et l'exécution normale du programme MIDA se poursuit.