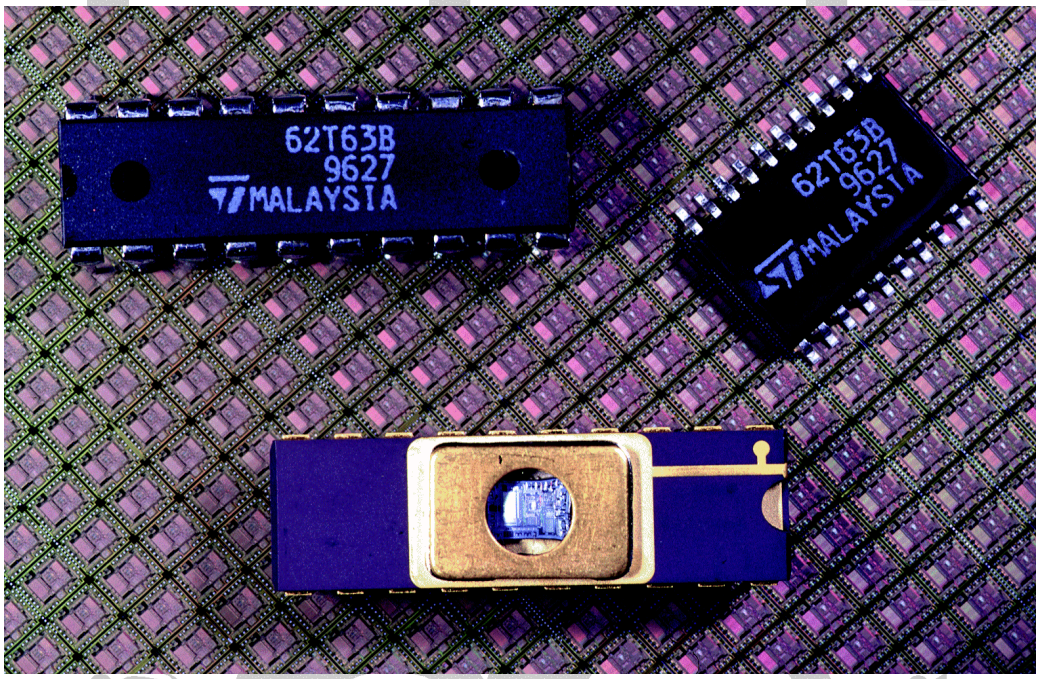
Les systèmes microprogrammés

Exemple du ST62



Date :14/01/01
Révision : 20
Fichiers : mpu1 ss exercices
CHOFFARDET / SERREAU

1 Généralités :

1.1 Logique programmée ou logique câblée ?

Le choix du type de logique pour résoudre un problème, dépend de plusieurs critères :
complexité ; coût ; évolutivité ; rapidité.

Complexité / Coût :
Dans la logique câblée, un problème est résolu par un ensemble de fonctions logiques, la taille du circuit croissant avec la complexité du problème.
La logique programmée nécessite un minimum de composant mais la taille du circuit n'augmente plus avec la complexité du problème (moins vite qu'en logique câblée).

En fait le coût des microcontrôleurs a tellement chuté que le seuil de rentabilité est très bas et peu de problèmes sont actuellement résolus en logique câblée.

Evolutivité :
En logique câblée, la moindre modification du problème entraîne la mise au point d'un nouveau circuit. Alors qu'en logique programmée on pourra parfois se contenter d'une modification du programme.

Rapidité :
En fait, la logique câblée garde l'avantage en terme de rapidité. On notera l'existence de circuits logiques programmables (Pal et GAL) à mi-chemin entre les deux technologies.

1.2 Historique du microprocesseur

Le microprocesseur est né en 1972 du développement de la technologie des semi-conducteurs.
La recherche avait rendu possible l'intégration de plusieurs milliers de transistors dans un seul et même boîtier. Le coût de développement de circuits aussi complexes, nécessaires en particulier pour la conquête spatiale, ne pouvait être amorti que sur de très grandes séries, mais utilisés en petites séries dans des applications différentes.
Le circuit logique programmable par l'utilisateur en fonction de l'application fût alors appelé microprocesseur.

Depuis, le microprocesseur a évolué dans trois directions :

- Ordinateurs : complexes et puissants, ils disposent de nombreux périphériques et d'une interface homme machine très évoluée, ces microprocesseurs sont très complexes.
- Automates programmables industriels (API) : ils sont orientés vers des applications industrielles et disposent donc d'interfaces de puissances permettant de commander différents actionneurs. Les microprocesseurs sont robustes et spécialisés dans des opérations de logique.
- Les microcontrôleurs : Il s'agit ici de mettre dans le même boîtier un nombre maximum de circuits afin d'obtenir un système performant mais peu encombrant.
  Dans le même boîtier on trouve le CPU, la mémoire RAM et ROM et les circuits d'entrée sortie.

Ce cours prendra comme exemple de système microprogrammé les microcontrôleurs fabriqués par ST MICROELECTRONICS utilisables avec le STARTER KIT ST626X.

1.3 Un microprocesseur pour quoi faire ?

De nos jours, les microprocesseurs sont partout :

A la maison :

Ordinateurs personnels ; machines à laver ; Fours à micro-ondes ; Aspirateur ; Climatisation ; Jouet ; Alarme.

Les voitures :

Airbag ; tableau de bord ; Allumage ; Carburation ; ABS

Industrie :

Les API ; machines outils ;

Le système microprogrammé doit pouvoir gérer des tâches très différentes :

Dialogue homme machine.
Contrôle d'énergie.
Surveillance de capteurs.
Cycles de fonctionnement.

Ils permettent d'améliorer :

Le facteur de puissance (compatibilité électromagnétique) ; le rendement .

Le programme gère ces tâches mais des informations doivent parvenir au système. Cela se fait par un ensemble d'entrées sorties qui peut être :

Entrées Tout ou Rien (TOR)
Entrées analogiques
Entrées numériques
Sorties tout ou rien
Sorties analogiques
Sorties numériques

La plupart des applications ci-dessus peuvent être réalisées avec des microcontrôleurs 8 bits, peu performants mais richement dotés de circuits d'entrées sorties. Ce marché est en pleine expansion.

1.4 un aperçu du marché

Le tableau suivant présente quelques caractéristiques de microcôntroleurs.

- Il ne s'agit que d'un aperçu du marché, chaque famille de microcontrôleur comporte quelques dizaines de produits. De plus certains produits récents ne sont pas présentés (AMTEL ; SCENIX). Les produits présentés ont été choisis car ils possèdent certaines caractéristiques communes (sortie PWM ; CAN ; liaison série).

- On ne peut se contenter de ce tableau pour faire une comparaison exhaustive de ces produits. Il manque : les outils de développement ; des caractéristiques matériels plus développés (interruptions ; pile ; jeu d'instruction) ; le coût d'un système de développement.

	Architecture	Puissance (MIPS)	ALU	ROM	RAM	E/S (A)	PWM	CAN	Périphériques	Boîtier	TIMER
Thomson ST6265	8bits	0,16	8 bits : + -	4 ko	256 o	21 (13)	1	1 8bits 70 µs	SPI ; EEPROM	DIP28	1 8 bits
Thomson ST72251G1	8 bits	2,7	8 bits : + - ×	4 ko	256 o	22 (6)	1	1 8 bits 7 µs	SPI ; I²C	DIP32	2 16 bits
Thomson ST9	8 / 16 bits	0,2	8 bits + ; - ;× ; ÷	16 ko	256 o	56	2	1 8 bits 7 µs	SCI ; SPI	PLCC 68	3 16 bits
Motorola 68HC05P8	8 bits	0,2	8 bits + ; -	4 ko	112 o	20	1			DIP 28	1 8 bits
Motorola 68HC11A7	8 bits	0,2	8 bits × ; ÷	8 ko	256 o	32 (8)	1	1 8 bits 16 µs	SPI ; SCI	PLCC 68	1 8 bits
PHILIPS 80C535	8 bits	1	8 bits +; -; × ; ÷	4 ko	128 o	32	1	1 bits 14 µs			2
Intel 80c196kc	16 bits	3	16 bits + - ×		256 o	(8)	3	10 bits	SPI		6
Microchip PIC16C74	8 bits	5	8 bits + ; -	4 ko	192	33 (8)	2	1 8 bits 1,6 µs	SPI ; I²C ; USART ; SCI	DIP40	2 8 bits 1 16 bits

On notera :
Le ST6265 est le moins puissant tant au niveau de son ALU que de sa puissance de calcul.
Le ST7 fait jeu égal avec le 80c196kc en 8 bits.
Les 68HC11 ; 80C535 et 80c196kc sont plus difficiles à mettre en œuvre (boîtier).
Le 68HC05 est basé sur un 6800 le 68HC11 sur un 6801.
Enfin le PIC16C74 est le plus puissant en 8 bits mais il n'a pas de multiplication câblée.

2 Structure matérielle d'un microcontrôleur

Synoptique des microcôntroleurs ST626X

Erreur! Aucune rubrique spécifiée.

- L'unité centrale exécute les opérations logiques ou arithmétiques qui permettent de calculer les données de sorties à partir des données d'entrées.
- les bus sont les liaisons électriques qui véhiculent les informations entre les différents composants du système électronique ;
- La mémoire stocke le programme, ainsi que toutes les données nécessaires au traitement.
- les interfaces d'entrées et sorties, permettent de communiquer avec les périphériques extérieurs au système
- L'horloge donne la cadence, par des impulsions électriques qui synchronisent le système.
- Un système d'interruptions qui permet la gestion d'événements inattendus
- Une alimentation

2.1 Unité centrale

Erreur! Aucune rubrique spécifiée.

L'unité centrale d'un microprocesseur comprend essentiellement :

- *Une unité arithmétique et logique* (UAL ou ALU) qui effectue les opérations arithmétiques (addition, soustraction, multiplication,...), les opérations logiques (ET, OU, COMPLEMENT...), et les tests ;
- *Un décodeur d'instructions* ou *séquenceur*, qui s'occupe de l'enchaînement des opérations pour réaliser les instructions du programme ;
- *Des registres* qui permettent de stocker des données, des instructions ou des adresses.

2.1.1 Unité arithmétique et logique

L'unité arithmétique et logique permet de faire les opérations demandées par le programme entre les opérandes (données d'entrées) pour obtenir un résultat (données de sortie).

Exemple : l'ALU du ST62

Il s'agit d'une ALU 8 bits, capable de traiter des octets. Elle peut exécuter :

- Des opérations arithmétiques Addition, Soustraction ;

- Des opérations logiques (ET, COMPLEMENT, Décalage à gauche).

Le déroulement d'une opération s'effectue de la façon suivante :

- L'unité arithmétique logique exécute l'opération et place le résultat obtenu dans l'accumulateur A (l'opérande d'entrée est alors effacé). L'unité arithmétique et logique place aussi certaines caractéristiques du résultat dans le registre d'état (résultat nul (Z), retenue (C)...) ;

- Le résultat contenu dans l'accumulateur A, peut maintenant être enregistré dans la mémoire.

C'est une ALU rudimentaire actuellement les ALU savent multiplier et diviser, faire des opérations logiques ET OU NON XOU sur des mots de 32 bits.

2.1.2 Décodeur d'instructions

Le décodeur d'instructions lit séquentiellement les instructions du programme dans la mémoire. Il les décode, et génère tous les signaux nécessaires pour l'exécution de chaque instruction.
En fonction des opérations demandées dans le programme, l'unité de contrôle commande tous les autres composants. Pour cette raison on l'appelle aussi *unité de commande*.

Son fonctionnement est complexe mais il est totalement transparent pour l'utilisateur d'un microprocesseur.

2.2 Liaisons

Les nappes de fils, les pistes du circuit imprimé ou les liaisons électriques dans le microcontrôleur véhiculant les informations entre les différents composants d'un système de traitement informatique, sont appelées bus.

Les signaux qui circulent dans un système de traitement sont :
- des données,
- des sélections d'adresse,
- des commandes.

Les bus correspondant à chaque type de signal sont respectivement :

- Le bus de données.
  En effet, pour communiquer avec d'autres systèmes, un microprocesseur doit envoyer ou recevoir des données c'est le rôle du bus de données le nombre de fils qui constituent ce bus est appelé largeur de bus. Un bus de données de 16 bits ou 32 bits par exemple
  Le bus de données du ST62 est un bus 8 bits.

- Le bus d'adresse.

Les données sont lues ou écrites à une adresse, la largeur de ce bus définit l'espace directement adressable par le microprocesseur. Ce bus peut être de 12 bits, 24 bits, 32 bits par exemple.
Le ST62 possède un bus 8 bits pour sa mémoire de données et de 12 bits pour sa mémoire de programme.

- Le bus de contrôle.

Enfin certains signaux sont nécessaires pour gérer ce travail le bus de contrôle qui comprend entre autres, les signaux suivants :

Lecture / écriture d'une donnée,
Horloge de synchronisation (pour coordonner les différents circuits),
Etat du microprocesseur (En attente, en interruption, en initialisation),
Signal de réinitialisation (RESET),
Etat du bus (libre, en lecture, en écriture),
Etc.

2.3 Stockage des Informations binaires

2.3.1 Fonction mémoire, adressage

La fonction mémoire est réalisée par tout dispositif capable d'**enregistrer**, de **conserver**, de **restituer** une information.
Une mémoire est un ensemble ordonné de registres qui contiennent des données numériques.
La capacité binaire de chaque registre de la mémoire définit la taille des données (4, 8, 16, 32 bits).
Pour accéder à toutes les données dans la mémoire, il faut adresser tous les registres en donnant à chacun un numéro appelé adresse.
Le nombre de registres ordonnés de la mémoire définit sa capacité en mots.
On peut actuellement fabriquer des mémoires qui ont une capacité de quelques millions d'octets (Mega-octets) en un seul boîtier de circuit intégré.

Exemple : mémoire de capacité 64 koctets
Chaque registre de la mémoire peut contenir des données numériques sur 8 bits (1 octet).
Le nombre de registre de la mémoire est de :
64 ko, soit 64×2^{10} octets = $64 \times 1024 = 65536$ octets
Chaque registre de la mémoire est repéré par une adresse :

- Le premier registre est repéré par l'adresse « 0 » ;
- Le registre suivant par l'adresse « 1 » et ainsi de suite ;
- La dernière adresse porte le numéro décimal « 65535 » soit « 1111 1111 1111 1111 » en binaire pur ou encore « FFFF » en hexadécimal.

On constate que l'adressage de 65536 registres nécessite 16 bits d'adresse puisque la plus grande adresse (65535) s'écrit sur 16 bits.

2.3.2 Ecriture ou lecture d'une donnée en mémoire

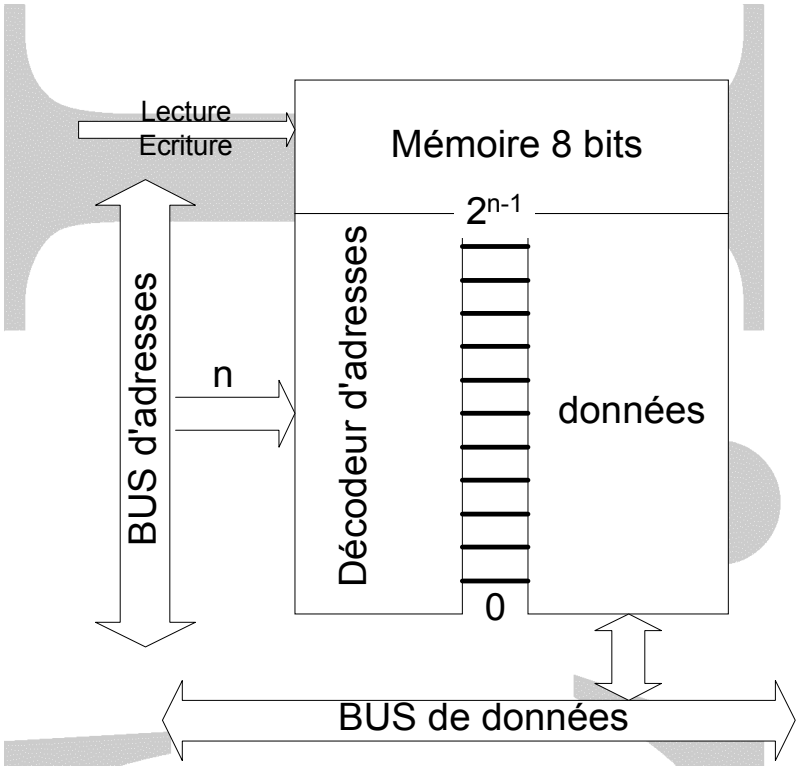
Pour écrire une donnée en mémoire le processeur exécute les opérations suivantes :

- Il sélectionne l'adresse choisie en plaçant sur le bus d'adresse sa valeur en binaire. Le décodeur d'adresse ouvre alors le registre sélectionné.

- Il place la donnée à mémoriser sur le bus de données.

- Il valide sur le bus de commande l'entrée « écriture ».

La donnée va alors s'écrire dans la mémoire à l'adresse sélectionnée.
Pour la lecture le principe est le même, seule la commande d'écriture est changée en commande de lecture.



2.3.3 Classement matériel des mémoires

On peut matériellement classer les mémoires en trois grandes familles : ROM, PROM, RAM.

■ Mémoire **ROM** *(Read Only Memory).*
La mémoire ROM est une mémoire morte à lecture seule. Son contenu est enregistré une fois pour toutes par le fabricant et ne peut plus jamais être modifié. Les coûts fixes de fabrication d'une telle mémoire sont importants et nécessitent une commande importante de plusieurs milliers d'unités. On peut comparer la ROM à un livre imprimé qui peut être lu autant de fois qu'on le souhaite mais qui ne peut pas être modifié.

■ Mémoire **OTP** *(One Time Programming)*
La mémoire OTP ou PROM est une mémoire à lecture seule, vierge à l'achat, et programmable une seule fois par l'utilisateur qui est le concepteur du système à microprocesseur. On peut comparer la l'OTP à un cahier sur lequel l'utilisateur inscrit à l'encre le programme.
Pour palier aux inconvénients de l'OTP (programmable une seule fois) Il existe des EPROM des EEPROM et des mémoire FLASH.

■ Mémoire **EPROM** *(Erase Programmable Read Only Memory).*
La mémoire EPROM est à lecture seule dans le système. Elle est vierge à l'achat, programmable par l'utilisateur mais éventuellement effaçable en totalité dans un effaceur d'EPROM par exposition aux rayons ultraviolets pendant une durée d'environ une demi-heure. Cet effacement est donc relativement long et doit être obligatoirement fait hors site.

■ Mémoire **EEPROM** *(Electrically Erasable Programmable Read Only Memory).*
La mémoire EEPROM est de même type qu'une mémoire EPROM mais effaçable électriquement en appliquant une tension sur certaines broches. L'effacement dans ce cas est beaucoup plus rapide (quelques secondes) et peut être fait avec le composant monté dans le système.

■ Mémoire **FLASH**

La mémoire Flash est de même type que L'EEPROM mais son effacement se fait par bloc, l'opération est donc beaucoup plus rapide.

■ Mémoire **RAM** *(Random Access Memory)*.

La mémoire RAM est une mémoire vive dans laquelle l'utilisateur peut écrire et lire à volonté. Cette mémoire perd son contenu en cas de coupure d'alimentation. Une pile de sauvegarde électrique est quelquefois associée à ce type de mémoire pour éviter un effacement intempestif dû à une coupure d'alimentation système.

2.3.4 Classement logiciel des mémoires

On peut aussi classer les mémoires par leur utilisation ou leur contenu : mémoire programme et données.

■ Mémoire programme

La mémoire programme contient la liste ordonnée des instructions à traiter par le processeur, c'est-à-dire le programme. Dans un système à microprocesseur la mémoire programme est en général de type ROM ou PROM.

Exemple : mémoire de Programme des ST6260 et ST6265

Il s'agit d'une mémoire 8 bits avec un bus d'adressage de 12 bits soit une capacité de 4 ko

Utilisation	Adresse
Réservé	0000h → 07Fh
Programme utilisateur pour les 6255 ; 6260 ; 6265 Réservé sinon	0080h → 087Fh
Programme utilisateur	0880h → 0F9Fh
Réservé	0FA0h → 0FEFh
Vecteur n°4 (CAN et Timer)	0FF0h → 0FF1h
Vecteur n° 3 (AR Timer)	0FF2h → 0FF3h
Vecteur n°2 Port C, SPI	0FF4h → 0FF5h
Vecteur n° 1 (Ports A et B)	0FF6h → 0FF7h
Réservé	0FF8h → 0FFBh
Vecteur n°0 (NMI)	0FFCh → 0FFDh
Vecteur Reset	0FFEh → 0FFFh

■ Mémoire de données

La mémoire de données sauvegarde les entrées en provenance des périphériques d'entrées (capteur, boutons de pupitre, clavier, ...), les résultats intermédiaires de calcul du processeur, les données de sorties (préactionneurs, voyants, afficheurs).

La mémoire de données est obligatoirement de type RAM puisque la lecture et l'écriture y sont nécessaires en permanence.

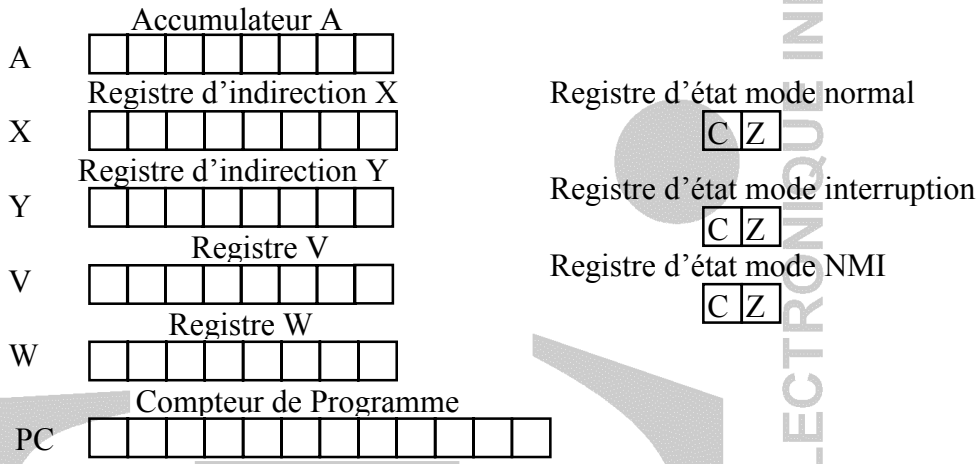
Exemple : mémoire de données des ST626X
Il s'agit d'une mémoire 8 bits avec un bus d'adresse de 8 bits soit une capacité de 256 octets

Contenu		Nom	Adresse
RAM ET EEPROM sélectionnée via le registre DRBR 64 octets par page	Mémoire		000h → 03fh
Fenêtre de données en ROM 64 octets	Mémoire		040h → 07fh
Registre X	CPU	X	080h
Registre Y		Y	081h
Registre V		V	082h
Registre W		W	083h
Données RAM 59 octets	Mémoire		084h → 0BFh
Registre de données du port A	E/S	DRA	0C0h
Registre de données du port B		DRB	0C1h
Registre de données du port C		DRC	0C2h
Réservé			0C3h
Registre de direction du port A	E/S	DDRA	0C4h
Registre de direction du port B		DDRB	0C5h
Registre de direction du port C		DDRC	0C6h
Réservé			0C7h
Registre des options d'interruptions	Interruptions	IOR	0C8h
Registre de la fenêtre de données en ROM	Mémoire	DRWR	0C9h
Réservé			0Cah → 0CBh
Registre des options du port A	E/S	ORA	0CCh
Registre des options du port B		ORB	0CDh
Registre des options du port C		ORC	0CEh
Réservé			0CFh
Registre de données du CAN	CAN	ADR	0D0h
Registre de contrôle du CAN		ADCR	0D1h
Registre prédiviseur du Timer 1	Temporisateur 1	PSC	0D2h
Registre compteur du Timer 1		TCR	0D3h
Registre d'état et de contrôle du Timer 1		TSCR	0D4h
Registre de contrôle du Timer à rechargement automatique	Temporisateur 2	ARMC	0D5h
Registre 1 d'état et de contrôle du Timer à rechargement automatique		ARSC0	0D6h
Registre 2 d'état et de contrôle du Timer à rechargement automatique		ARSC1	0D7h
Registre du chien de garde		WDR	0D8h
Registre de rechargement et de capture du Timer à rechargement automatique	Temporisateur 2	ARRC	0D9h
Registre de comparaison du Timer à rechargement automatique		ARCP	0DAh
AR Timer load register		ARLR	0DBh
Registre de contrôle de l'oscillateur	Oscillateur	OSCR	0DCh
Registre divers	SPI	MR	0DDh
Réservé			0DEh → 0DFh
Registre de données de l'interface série	SPI	DSR	0E0h
Registre diviseur de l'interface série		SPIDIV	0E1h
Registre du mode de l'interface série		SPIMOD	0E2h
Réservé			0E3h →0E7h
Registre de données RAM/EEPROM	EEPROM	DRBR	0E8h
Réservé			0E9h
Registre de contrôle de l'EEPROM	EEPROM	EECTL	0EAh
Réservé			0Ech → 0FEh
Accumulateur	CPU	A	0FFh

Les registres en rouge ne doivent pas être adressés par les instructions SET RES

2.3.5 Registres internes

Les registres peuvent être plus ou moins nombreux dans l'unité centrale.
Un grand nombre de registres internes permet un maximum du traitement des données à l'intérieur de l'unité centrale. On évite ainsi de nombreux accès du processeur à la mémoire pour stocker et lire des données intermédiaires de calcul. Le temps de traitement se trouve ainsi réduit.
Un minimum de registres internes est indispensable au fonctionnement de l'unité centrale et nécessaire à la programmation.



Exemple : inventaire des registres du ST62XX

- L'accumulateur A contient les opérandes traités par l'unité arithmétique et logique.
- Les registres X et Y sont des registres d'indirection, utilisés pour l'adressage indirect.
- Les registres V et W sont des registres auxiliaires.
- Le registre d'état CC contient des indications précieuses sur le résultat après chaque traitement par l'unité arithmétique et logique :
  - le bit zéro Z est mis à un quand le résultat de l'opération est zéro,
  - Le bit de retenue C est mis à un quand le résultat ne peut tenir dans l'accumulateur de sortie A, c'est à dire s'il y a débordement.

  Il y a trois registres d'état contenant les bits Z et C. Un pour chaque mode de fonctionnement du microprocesseur (Normal, Interruption, Interruption non masquable).
- Le compteur ordinal PC est le registre qui pointe dans la mémoire l'adresse de la prochaine instruction à traiter par l'unité de commande. Il s'agit d'un registre 12 bits, capable d'adresser 4 ko.

En fait, seul les registres CC ; PC et la pile sont des registres internes Les registres A, X, Y, V, W sont présents dans la mémoire de données et sont adressés par des modes particuliers d'adressage (direct court).

La valeur des registres d'un microprocesseur peut être incrémentée ou décrémentée directement par l'unité de contrôle.

2.3.6 L'EEPROM et la fenêtre de données en RAM

L'architecture du ST62 sépare mémoire de données et mémoire programme. Cela a l'avantage d'accélérer le traitement des données car on accède simultanément à ces deux zones.
Cependant, il n'est pas possible d'accéder à des tables de données en ROM (la mémoire de données est de type RAM) et l'espace données est restreint : 256 octets auxquels il faut enlever tous les registres des différents périphériques du ST62.

Ces deux problèmes ont étés contournés de la façon suivante :

1/ On a utilisé le principe des bancs de mémoire. Ainsi la zone mémoire 00h -> 3fh contient 64 octets qui sont soit de la RAM, soit une EEPROM elle-même divisée en deux (pour les ST62 dotés de 128 octets d'EEPROM). Le choix du type de mémoire se fait par l'intermédiaire du registre de données RAM/EEPROM DRBR suivant le tableau ci dessous

☞ **DRBR :** Registre du banc de données RAM/EEPROM *(Data RAM/EPROM Bank Register).*
Adresse : E8h - Ecriture seule.

DRBR	ST6253 / 52 / 55	ST6260 / 65	ST6263 / 62
00	Rien	Rien	Rien
01	Non disponible	EEPROM page 0	EEPROM page 0
02	Non disponible	EEPROM page 1	Non disponible
08	Non disponible	Non disponible	Non disponible
10h	RAM page 2	RAM page 2	RAM page 2
Autre	Réservé	Réservé	Réservé

Ce registre ne peut être qu'écrit, on ne doit donc pas utiliser les instruction SET / RES pour le modifier.

2/ il est possible de faire apparaître une donnée de la mémoire de programme dans la mémoire de donnée. En fait, ce sont 64 octets de la mémoire de programme qui sont vus dans la fenêtre de données en RAM (zone 084h → 0BFh)

Le choix de la zone vue en RAM se fait par le registre DRWR (DATA ROM WINDOW REGISTER).

☞ **DRWR :** Registre de la fenêtre de données en RAM *(Data ROM Window Register).*
Adresse : C9h – Ecriture seule.

Une adresse de la mémoire programme est sur 12 bits, comme 64 octets sont vus (6 bits), il faut préciser les 6 premier bits de l'adresse dans le registre DRWR (Les bits 6 et 7 seront ignorés).
Ainsi, lors de l'accès à une donnée dans cet espace seul les 6 premiers bits de l'adresse de la donnée seront pris en compte pour obtenir le reste de l'adresse, les deux bits de poids fort seront ignorés.

Synoptique de l'espace mémoire du ST62 :

Erreur! Aucune rubrique spécifiée.

Exercice : Fenêtre de données en RAM.

On a pré calculé une table de sinus de 64 valeurs reparties de 0 à $\frac{\pi}{4}$. On souhaite qu'elle soit accessible en mémoire de données.

- Proposer un emplacement de cette table en mémoire programme.
- Indiquer la valeur du registre DRWR pour accéder à cette table.
- Indiquer l'adresse en RAM des valeurs de $0 ; \frac{\pi}{4} ; \frac{\pi}{6}$

2.4 Interfaçage

2.4.1 Introduction :

Un système de traitement à microprocesseur communique avec les périphériques d'entrées sorties par écriture ou lecture de valeurs numériques binaires (données codées).
L'interface est la fonction qui permet de transférer les données entre le système de traitement et un périphérique extérieur (imprimante, pré actionneurs, clavier, capteur, boutons, ...).

Erreur! Aucune rubrique spécifiée.

Le microprocesseur écrit ou lit le contenu du registre de données dans l'interface qui se charge de la communication avec le périphérique.
Le microprocesseur configure le périphérique par des registres de commandes. Ces registres de commandes, situés physiquement dans les circuits d'interface, sont vus par le microprocesseur comme des registres de la mémoire de donnée.

2.4.2 L'EEPROM

Caractéristiques :

300 00 cycles d'écriture ; temps de rétention de 10 ans
Lecture transparente pour l'utilisateur
Temps d'écriture : 5 ms
2 modes d'écriture :
- Octet par octet
- 8 octets consécutifs

Organisation :

L'EEPROM est sélectionnée via le registre DRBR elle est organisée en 8 colonnes de 8 octets :

Colonne 0 → octets 0 à 7
Colonne 1 → octets 8 à 15
Etc …
Colonne 7 → octets 56 à 63
Les procédures de lecture, écriture et écriture parallèle sont les suivantes :

Erreur! Aucune rubrique spécifiée.

Registre :

L'écriture en EEPROM passe par le registre de contrôle de l'EEPROM EECTL

☞ **EECTL** : Registre de contrôle de l'EEPROM *(EEPROM Control Register)*.
Adresse : EAh – Lecture/ecriture

D7	E2OFF	D5	D4	E2PAR1	E2PAR2	E2 BUSY	E2ENA

➥ E2OFF : Désactivation de l'EEPROM (Ecriture seule).
Permet de désactiver l'EEPROM si ce bit est activé, tout accès sera sans effet et la consommation de L'EEPROM réduite au minimum.

➥ E2BUSY : EEPROM occupée (lecture seule).
Indique que l'EEPROM est occupée, toute tentative d'accès à l'EEPROM lorsque ce bit est activé sera sans effet et le cycle d'écriture se terminera normalement.
Il faut tester ce bit avant tout accès à l'EEPROM.

➥ E2ENA : Autorisation de l'EEPROM (Ecriture seule).
Ce bit autorise la programmation de l'EEPROM. Il doit être activé avant toute tentative d'écriture.

Il est également possible d'écrire jusqu'à 8 octets d'une colonne en même temps par l'utilisation des Bits E2PAR1 et E2PAR2. ce mode d'écriture est appelé mode parallèle.

2.4.3 Port d'entrées sorties

Caractéristiques :
Le ST62 possède trois ports 8 bits d'entrées sorties notés PA, PB, PC.
Chaque bit d'un port peut être configuré individuellement en entrée ou en sortie.
Les données à écrire ou à lire sont lues ou écrites sur les registres DRA DRB ou DRC.
La configuration des ports se fait par les registres ORX, DDRX et DRX lorsque le port est configuré en entrée (X=A, B ou C).

Schéma interne d'une entrée /sortie du ST62

Erreur! Aucune rubrique spécifiée.

Exercice : Registre des ports D'E/S.
Compléter ce tableau à partir du schéma précédent

DDRX	ORX	DRX	Description
			Entrée avec polarisation, sans interruption (état à l'initialisation)
			Entrée sans polarisation
			Entrée avec polarisation et interruption
			Entrée analogique
			Sortie à drain ouvert
			Sortie Push-pull

Mise en œuvre :
Lorsque l'on modifie un port d'entrées sorties, on ne peut pas modifier les trois registres en même temps. On passe donc par des configurations intermédiaires.
Le diagramme suivant donne les transitions possibles d'un port. Ne pas suivre ce diagramme peut entraîner des dysfonctionnements du microcontrôleur.

La configuration est donnée dans l'ordre : DDRX, ORX, DRX

Erreur! Aucune rubrique spécifiée.

Exercice : Configuration des registres DDRX, DRX et ORX.
On souhaite configurer les ports d'entrées sorties du ST6260 de la façon suivante :

Port A : PA0 à PA3 Sorties Push Pull.
Port B : PB0 à PB2: entrées sans polarisation.
Port C : PC2 entrée analogique PC3 entrée avec polarisation et interruption.

- Donner la valeur des registres DRX ; ORX ; DDRX correspondant à cette configuration.
- Proposer un algorithme de programmation de ces registres.

2.4.4 Interface série (SPI)

On se contentera ici de décrire le matériel de la SPI la mise en œuvre logiciel (protocole, contrôle de flux) dépassant l'objet de ce cour.
A partir de la SPI, il est également possible d'implémenter un bus I2C, une UART par voie logicielle. La note d'application AN914 *"Using the ST626X SPI as UART"* décrit cette mise en œuvre.

Caractéristiques :

- Registre à décalage 8 bits,
- Nombre de bits transmis programmable,
- Taux de transfert de 2400 à 61530 Baud
- Possibilité de mettre en œuvre différents protocoles (RS232, I²C, …)

Synoptique de la SPI :

Erreur! Aucune rubrique spécifiée.

Fonctionnement :

Les données sont envoyées ou reçues sur deux lignes la SPI se charge de replacer les bits dans son registre à décalage de la SPI (SPIDSR).
L'ensemble est synchroniser par un signal d'horloge émis sur la ligne SCK.

Registres :

La SPI est utilisée via 4 registres MOD ; DIV ; SPIDSR ; MISC

☞ **MOD :** registre de contrôle du SPI *(SPI Mode Control Register).*
Adresse : E2h – Lecture / écriture.

SPRUN	SPIE	CPHA	SPCLK	SPIN	SPSTRT	EFILT	CPOL

➥ SPRUN : SPI RUN
Il s'agit d'un drapeau d'activité de la SPI. Il peut être utilisé en émission comme en réception. Il est automatiquement remis à 0 zéro à la fin d'une réception ou d'une émission par la SPI.

➥ SPSTRT : START SELECTION
Ce bit configure la façon dont commence une émission ou une réception.
Si SPSTRT = 0 l'émission ou la réception commence avec la mise à 1 de SPRUN
Si SPSTRT = 1 l'émission ou la réception commence lorsque SPRUN = 1 ET Sin = 1

➥ SPIE : Autorisation des interruptions SPI
Lorsque ce bit est à 1, les interruptions du SPI sont autorisées.

➥ SPCLK : Choix de la source pour l'horloge.
Si SPCLK = 0, SCK doit être configuré comme une entrée. L'horloge est externe et la SPI en mode esclave.
Si SPCLK = 0, SCK doit être configurée comme une sortie. L'horloge est alors interne. La SPI est alors en mode maître. La phase et la polarité de l'horloge sont alors configurées par CPHA et CPOL

➥ CPHA : et CPOL : voir les chronogrammes du data book.

➥ EFILT : Activation des filtres.
Si EFILT = 1 les filtres d'immunité au bruit sont activés sur Sin et SCK.

SPIN : Si SPIN = 0 l'entrée de SPIDSR = 0
Si SPIN = 1 l'entrée du SPIDSR est relié à l'entrée Sin

☞ **DIV** : Registre de division *(SPI DIVider Register).*
Adresse E1h – Lecture écriture.

SPINT	DIV6	DIV5	DIV4	DIV3	CD2	CD1	CD0

➥ SPINT : Interruption SPI (Lecture seule).
Drapeau d'interruption SPI. Une interruption est générée à la fin d'une réception ou à la fin d'une émission. Il doit être remis à zéro par le logiciel.

➥ CD2-CD0 : Choix du facteur de division.

Sélectionne la valeur du diviseur de l'horloge de synchronisation présente sur SCK en mode maître.

CD2	CD1	CD0	Facteur de division	BAUD (f_{osc}=4MHz)	BAUD (f_{osc}=8MHz)
0	0	0	1	307600	615300
0	0	1	2	153800	307600
0	1	0	4	76800	153800
0	1	1	8	38400	76800
1	0	0	16	19200	38400
1	0	1	32	9600	19200
1	1	0	64	2400	9600
1	1	1	128	1200	2400

➥ DIV6-DIV3 : Nombre de bits transmis

Le tableau ci dessous indique le nombre de bits transmis en fonction de DIV6 – DIV3

Il n'est pas conseillé d'envoyer plus de 8 bits.

DIV6	DIV5	DIV4	DIV3	Nombre de bits
0	0	0	0	Réservé
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

☞ **MISC** : Registre Divers *(MISCelanous register).*

Adresse : DDh – Lecture – écriture.

Seul le bit 0 de ce registre est utilisé lorsqu'il est à 1, il configure Sout comme la sortie de la SPI.

☞ **SPIDSR** : Registre de données de la SPI *(SPI Data Shift Register).*

Adresse E0h – Lecture écriture

La

2.4.5 Temporisateur

Un Timer est un périphérique qui permet de générer des impulsions de durée choisie, de mesurer des temps, de provoquer des événements à des instants choisis.

Caractéristiques :

- Décompteur 8 bits
- Prédiviseur 7 bits programmable
- 1 interruption masquable (fin de comptage)
- Entrée temporisateur externe
- 3 modes de fonctionnement
  - Mesure d'impulsion
  - Compteur d'éventements
  - Générateur d'impulsion
- Le temporisateur peut être arrêté par logiciel

Synoptique du Temporisateur :

Erreur! Aucune rubrique spécifiée.

Fonctionnement :

Le prédiviseur :

Le prédiviseur peut être cadencé soit par l'oscillateur principal (f_{osc} / 12) soit par une horloge externe sur l'entrée TIMER.
Le prédiviseur est décrémenté sur chaque fronts montant de son entrée de comptage.
La sortie est multiplexée ce qui permet un facteur de division de 1 à 128.

La sortie du prédiviseur décrémente le compteur 8 bits.
Le registre PSC peut être lu et écrit sans restriction. Le bit 7 n'est pas utilisé et sa lecture donnera toujours 1.

Le compteur :

Le compteur est décrémenté à chaque front montant de la sortie du prédiviseur.
Le bit TMZ (TiMer Zero) est placé à 1 lorsque le compteur arrive à 0.
Le Bit ETI permet d'autoriser une interruption Timer (TMZ ET ETI).
Le TCR continue automatiquement son décomptage de 0ffh.

Registres :

Le temporisateur est utilisé via trois registres :
PSC : registre du prédiviseur
TCR : registre du compteur
TSCR : registre de configuration et de contrôle

☞ **TSCR** Registre d'état et de contrôle du temporisateur *(Timer Status Control register).*
Adresse : D4h – Lecture écriture.

TMZ	ETI	TOUT	DOUT	PSI	PS2	PS1	PS0

➥ PS0-PS1
Ces trois bits règlent le multiplexeur 8 vers 1.

PS2	PS1	PS0	Diviseur
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

➥ PSI : Bit d'initialisation du prédiviseur.
Si PSI = 0 le prédiviseur est chargé avec 7fh et le compteur est bloqué.
Si PSI = 1 le décomptage est autorisé.

➥ TOUT, DOUT : Contrôle du mode de fonctionnement du TIMER

TOUT = 0 ; DOUT = 1 : mesure d'impulsion.
Dans ce mode le prédiviseur est décrémenté par $f_{osc}/12$ mais uniquement si l'entrée Timer est à 1.

TOUT = 0 ; DOUT = 0
Dans ce mode, la broche TIMER est une entrée et le prédiviseur est décrémenté sur les fronts montants de cette entrée. Cela permet de compter les événements survenant sur cette entrée.

TOUT = 1, TIMER = DOUT
Dans ce mode, la broche TIMER est connectée à DOUT, le prédiviseur est cadencé par $f_{osc}/12$.
Le contenu de DOUT est envoyé vers la sortie TIMER lorsque TMZ = 1.

➥ ETI : Autorisation des interruptions du temporisateur.
Si ETI = 1, les interruptions du Temporisateur sont activées.

➥ TMZ : Temporisateur à 0.
TMZ = 1 lorsque TCR = 0h

.

☞ **TCR** : Registre du temporisateur *(Timer Counter Register)*.
Adresse : D3h – Lecture écriture.

☞ **PSC** : Registre du prédiviseur *(PreSCaler Register)*.
Adresse : D2h – Lecture écriture.

Note :
La broche TIMER n'est présente que sur les microcontrôleurs 6265 et 6255.
Lorsque cette broche n'est pas présente, l'intérêt de ce temporisateur est fortement diminuée.
Les bits DOUT et TOUT ne sont pas utilisés. TOUT doit toujours être à 1.

Exercice : Temporisateur

On souhaite générer des interruptions périodiques (passage à zéro du temporisateur). Quelles sont les fréquences possibles ?

Le Timer 1 possède un prédiviseur par 1 ; 2 ; 4 ; 8 ; 16 ; 32 ; 64 ; 128. C'est un compteur 8 bits
Quelle est la longueur maximum de l'impulsion que l'on peut générer ?

Quelle est la résolution minimum de la mesure d'une impulsion ?

2.4.6 Temporisateur à rechargement automatique

Le Timer à rechargement automatique permet de créer des événements périodiques en particulier un signal PWM.

Caractéristiques :

- Compteur 8 bits avec un prédiviseur programmable,
- Fréquence maximum : f_{osc},
- 1 ligne d'interruption masquable,
- 4 modes de fonctionnement :
  - Rechargement (PWM et base de temps),
  - Compteur d'événements,
  - Mesure de période,
  - Boucle à verrouillage de phase.

Synoptique du temporisateur à rechargement automatique :

Erreur! Aucune rubrique spécifiée.

Fonctionnement :

Prédiviseur :
Le prédiviseur peut être cadencé par trois sources : L'oscillateur ; l'oscillateur divisé par 3 ou une source extérieure sur ARTIMin.
Le choix du facteur de division se fait par les bits PS0 – PS2.
La sortie du prédiviseur est reliée au compteur ARTC.

Le compteur :
Lorsque le compteur déborde, son contenu est automatiquement rechargé avec la valeur du registre ARRC. La sortie est activée (ARTIMout =1 si PWMOE =1). Une interruption est générée (drapeau OVF) si OVIE= 1.
Lorsque ARTC = ARCP la sortie est désactivée (ARTIMout = 0 si PWOE = 1)
Une interruption est générée (drapeau CPF) si CPIE = 1.

Registres :

Il ne faut pas moins de 6 registres pour l'utiliser :

☞ **ARSC1** : Registre d'état et de contrôle du temporisateur à rechargement automatique *(Auto Reload Status/control Register).*
Adresse : D7h – Lecture écriture.

PS2	PS1	PS0	D4	SL1	SL0	CC1	CC0

Ce registre configure le prédiviseur.

➥ CC1-CC0 : choix de la source du prédiviseur.
Il est cadencé soit par f_{osc} soit par $f_{osc}/3$ soit par un signal externe appliqué sur ARTIMin.

CC1	CC0	Horloge
0	0	f_{osc}
0	1	$f_{osc}/3$
1	0	ARTIMin
1	1	Réservé

➥ SL0 : Validation de l'entrée ARTIMin
L'entrée ARTIMin est validée si SL0 = 1.

➥ SL1 : Front actif.
SL1 permet de choisir un déclenchement sur front montant (SL1 = 0) ou sur front descendant (SL1 = 1).

➥ PS2-PS0 : Facteur de division du prédiviseur.

PS2	PS1	PS0	Diviseur
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

☞ **ARMC** Registre de contrôle du temporisateur *(Auto Reload Mode Control Register).*
Adresse D5h – Lecture / écriture

TLCD	TEN	PWMOE	EIE	CPIE	OVIE	ARMC1	ARMC0

➥ TLCD : Chargement du temporisateur.
lorsque ce bit est activé le contenu du registre ARRC est copié dans le compteur ARC. Le registre ARPSC est initialisé à 0.

➥ TEN : Validation de l'horloge
La mise à 1 de ce bit lance le prédiviseur et donc le compteur.

➥ PWMOE : Activation de la sortie PWM.
La mise à 1 de ce bit active la sortie ARTIMout

➥ ARMC1 – ARMC0 : Mode de fonctionnement du temporisateur.
Les bits ARMC1 et ARMC0 configurent le temporisateur suivant le tableau ci dessous.

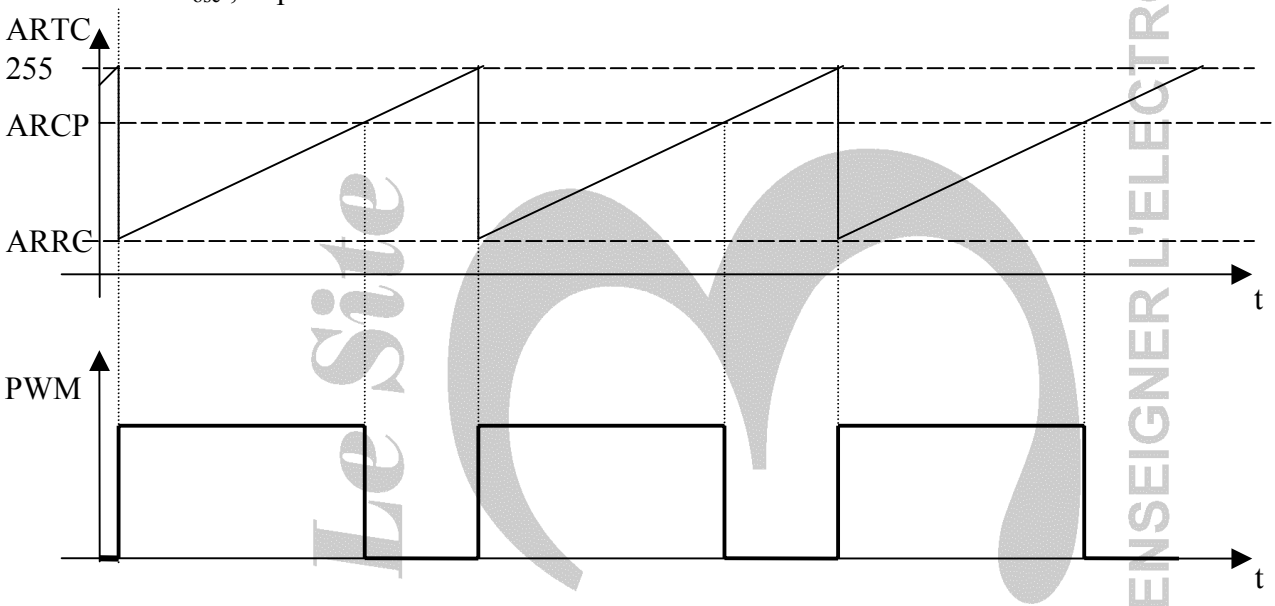
ARMC1	ARMC0	Mode de fonctionnement
0	0	Rechargement automatique
0	1	Mode capture
1	0	Mode capture avec remise à zéro de ARTC et ARPSC
1	1	Chargement sur un front extérieur

Mode PWM

Lorsque le compteur ARTC déborde (OVF) le compteur est automatiquement rechargé avec la valeur du registre ARRC et la sortie ARTIMout est forcée à 0.

Lorsque le compteur atteint la valeur du registre ARCP, la sortie ARTIMout est forcée à 1.

La valeur du rapport cyclique est fixée par ARCP sa résolution par 255 – ARRC.
La fréquence par :
F_{osc} ; le prédiviseur et 255 – ARRC



Mode Capture

Dans ce mode, l'opération de capture (transfert du registre ARRC vers ARTC) est effectuée lors d'un front actif sur l'entrée ARTIMin.

Mode Capture avec remise à zéro de ARTC et ARPSC.
Ce mode est identique au précédant mais les registres ARTC et ARPSC sont remis à zéro.

➥ EIE - CPIE - OVIE : Autorisation des interruptions
Ces bits autorisent les interruptions correspondant aux drapeaux EF ; CPF et OVF du registre ARSC0.

☞ **ARSC0** : Registre de contrôle et d'état du temporisateur *(Auto Reload Status Control Register 0)*.
Adresse : D5h – Lecture / écriture.

D7	D6	D5	D4	D3	EF	CPF	OVF

Les bits D7 à D3 ne sont pas utilisés.

➥ EF : Drapeau d'interruption externe.
Ce bit passe à un lorsqu'un front actif est présent sur l'entrée ARTIMin.

➥ CPF : Drapeau d'interruption comparaison.
Ce bit passe à 1 lorsque ARC et ARCP sont égaux.

➥ OVF : Drapeau d'interruption débordement.
Ce bit passe à 1 lorsque ARC passe de 0ffh à 0h.

☞ **ARLR** : Registre de chargement du temporisateur *(AR Load Registre)*.
Adresse : DBh – Lecture / écriture.
Ce registre permet de lire et écrire le registre ARTC.

☞ **ARRC** : Registre de rechargement et de capture *(AR Reload/Capture Register)*.
Adresse : D9h – lecture écriture.
Ce registre est transféré dans le compteur lors d'un débordement.

☞ **ARCP** : Registre de comparaison *(AR Compare Register)*.
Adresse D9h – Lecture écriture.
Il s'agit du registre de comparaison.

Exercice : Configuration du temporisateur à rechargement automatique :

On souhaite générer un signal PWM de fréquence 5 kHz avec un ST6260 cadencé à 8 MHz.

- Donner la valeur des registres et proposer un organigramme d'utilisation du temporisateur.

Le Timer à rechargement automatique (ARTIMER) est capable de générer un signal carré de rapport cyclique variable. α peut prendre 256 valeurs. Quelle est la fréquence maximale du signal ?

2.4.7 Convertisseur Analogique Numérique.

Caractéristiques :
- Conversion par approximations successives.
- Résolution de 8 bits.
- Erreur inférieure à + ou – 2 LSB.
- Temps de conversion : 70 µs à 8MHz.
- Interruption générée à la fin de la conversion.
- Possibilité de le désactiver par voie logicielle afin de réduire la consommation.

Synoptique du Convertisseur analogique numérique :

Erreur! Aucune rubrique spécifiée.

Fonctionnement :
Il s'agit d'un convertisseur par approximations successives. Il est activé par la mise à 1 du BIT PDS. La conversion est lancée par la mise à 1 du bit STA. Lorsque la conversion est terminée, EOC passe à 1 et une interruption est générée si EAI = 1.
Le résultat peut être lu dans le registre ADR. La mise à 1 du bit STA force à 0 le bit EOC.

Mise en œuvre :
- La précision du CAN décroît en dessous de 1,2 MHz (à 32 kHz la précision n'est plus que de + ou – 4 LSB)
- Ne jamais configurer plus d'une entrée en tant qu'entrée analogique.
- Laisser au moins une instruction entre l'activation du CAN et le début d'une conversion.
- Pour un meilleur résultat : placer le processeur en attente (Mode WAIT) ; autoriser les interruptions du CAN et éviter de modifier les sorties.
- La capacité d'entrée du convertisseur est d'environ 12 pF. L'impédance du signal mesuré ne doit pas dépassé 30 kΩ afin que cette capacité soit correctement chargée.

Registres :

☞ **ADCR :** Registre de contrôle du CAN (Analog Digital Control Register).
Adresse : D1h – Lecture écriture.

EAI	EOC	STA	PDS	N.A	N.A	N.A	N.A

➥ EAI : Autorisation de l'interruption CAN
Lorsque ce bit est mis à 1 il autorise les interruptions du CAN.

➥ EOC : Fin de conversion
La fin de la conversion est signalée par le passage à 1 de ce bit. Il est automatiquement remis à 0 lorsqu'une nouvelle conversion commence.
Si l'interruption CAN est activée, une interruption est déclenchée (vecteurn°4) lorsque ce bit passe à 1.

➥ STA : Début de la conversion.
La conversion est lancée lorsque ce bit est mis à 1.

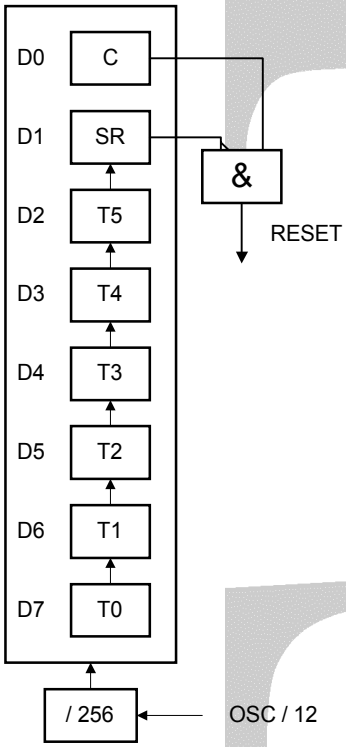
➥ PDS : Mise en veille.

Ce bit doit être mis à 1 afin d'activer le CAN, sa mise à 0 permet de diminuer la consommation du ST62.

☞ **ADR** : Résultat de la conversion *(Analog Digital Register)*.
Adresse : D0h – Lecture / écriture
Le résultat de la conversion est lue dans ce registre.

Exercice : CAN.
Donner l'algorithme de l'utilisation du CAN sans interruption.

2.4.8 Chien de garde



Le chien de garde est en fait un compteur qui est régulièrement décrémenté. Arrivé à 0, il provoque une interruption. Il doit donc être régulièrement remis à sa valeur maximum. Cela permet de surveiller le fonctionnement du microcontrôleur et d'éviter qu'il ne se bloque.
Le compteur du chien de garde est le registre WDR.

☞ **WDR** : Registre du chien de garde *(Watch Dog Register)*.

T0	T1	T2	T3	T4	T5	SR	C

➥ C : contrôle du chien de garde :

Il existe deux types de chien de garde Hardware et software le type de chien de garde est choisi lors du choix des options de la programmation du microcontrôleur.
Lorsque le chien de garde est hardware, le bit c est forcé à 1, le chien de garde est toujours actif.
Lorsque le chien de garde est software, il faut forcer le bit C à 1 afin d'activer le chien de garde.
A ce moment là, on ne peut plus le désactiver.

Si le bit C est à 0, le chien de garde peut alors être utilisé comme un Timer 7 bits.

➥ SR : Bit de reset logiciel :

Lorsque C = 0 : il s'agit du bit de poids le plus fort (MSB) du Timer 7 bits
Lorsque C = 1 : la mise à 0 de ce bit provoque un Reset.

➥ T5-T0 : compteur

T5 est le bit du compteur le plus significatif (MSB) et T0 le moins significatif (LSB).

En fait, SR le bit T6 du compteur (décompteur) du chien de garde.
Son passage à zéro déclenche le reset (si C = 1).

Exercice : Chien de garde.
Combien de cycle mini et maxi peut-on régler avec le chien de garde ?

2.4.9 Conclusion

Le ST62 possède, plus d'entrées sorties que de broches sur son boîtier, certaines broches ont donc un rôle multiple. Par exemple on ne peut pas utiliser PB7 et la sortie du Timer à rechargement automatique. D'autres sorties ne sont pas disponibles (sorties PC4 à PC7 sur le ST6260 par exemple).

2.5 l'Horloge

L'horloge synchronise tous les circuits du microcontrôleur. Plus elle est rapide et plus le microcontrôleur va vite. En contre partie il consomme plus.

Synoptique des différentes horloges du ST62 :

Erreur! Aucune rubrique spécifiée.

Un oscillateur auxiliaire basse fréquence permet l'utilisation d'un oscillateur interne par la mise à un du bit OSCOFF du registre ADCR.

Un oscillateur de secours (OSG) permet de filtrer le signal de l'oscillateur principal, limite la fréquence interne (en sortie du diviseur par 1, 2 ou4) donne la main à l'oscillateur auxiliaire basse fréquence (LFAO) en cas de coupure de l'oscillateur principal.

☞ **OSCR** : Registre de l'oscillateur *(OSCillator Register)*.
Adresse DCh – Ecriture seule.

-	-	-	-	OSCR3	OSCR2	RS1	RS0

➥ RS0 - RS1 : Diviseur.

RS1	RS0	Diviseur par
0	0	1
0	1	2
1	0	4
1	1	4

➥ OSRC3
Ce bit doit être activé afin de minimiser la consommation du ST62.

➥ OSRC2
Doit rester à 0.

Exercice : Horloges.

Quelle est la durée d'un cycle du CPU du ST62 (f_{osc} = 8 MHz) ?

T =

Une instruction du cpu dure en général 4 cycles

Quelle est la durée moyenne d'une instruction ?

$T_{instruction}$ =

Combien le CPU effectue-t-il d'instructions par seconde ?

N =

2.6 Les interruptions

L'étude des interruptions est complexe et dépend largement du microprocesseur. On se limitera aux principes communs à la gestion des interruptions ainsi qu'aux exemples nécessaires à la compréhension générale du ST62.
La programmation par interruption est un domaine d'étude appelé "programmation temps réel"

Un microprocesseur doit pouvoir interrompre son travail pour exécuter des opérations dont le traitement doit être immédiat (surcharge, arrêt d'urgence par exemple). Il doit éventuellement être capable de reprendre le cours de son travail après le traitement.
Ce traitement particulier s'appelle interruption. Il existe plusieurs types d'interruptions.

Quelques exemples :

Demande de réinitialisation (RESET),
Evénement extérieur demandant un traitement immédiat,
Fin d'une conversion analogique numérique.

Les sources d'interruption :

Les sources d'interruption peuvent être internes (CAN, Timer) ou externes (Reset, événement extérieur) lorsqu'elles sont externes, une broche du circuit est utilisée comme source d'interruption.
Les interruptions sont hiérarchisées (priorités). En effet, une demande de RESET par exemple ne doit pas être interrompue par la fin d'une conversion.

Les sources d'interruption du ST62 sont les suivantes :
PortA ; PortB ; ARTimer ; Timer A ; SPI ; CAN ; entrée NMI ; RESET.

Le ST62 distingue deux niveaux d'interruptions : les Interruptions non masquables et les interruptions classiques.

Interruptions non masquables : RESET et l'interruption NMI (interruption prioritaire)
Interruptions classique : CAN, Timer, Port d'entrée sortie, etc. (interruption non prioritaire)

Vecteurs d'interruptions :
Ces différentes sources d'interruptions demandent des traitements différents (différents programmes). La liste des emplacements de ces programmes est appelée **table des vecteurs d'interruption**. Un vecteur d'interruption est un emplacement particulier de la mémoire qui contient l'adresse du programme d'interruption. Parfois, plusieurs sources se partagent le même vecteur (la même adresse). C'est alors au programme de rechercher la source et d'effectuer un traitement conditionnel.

Les vecteurs du ST6260

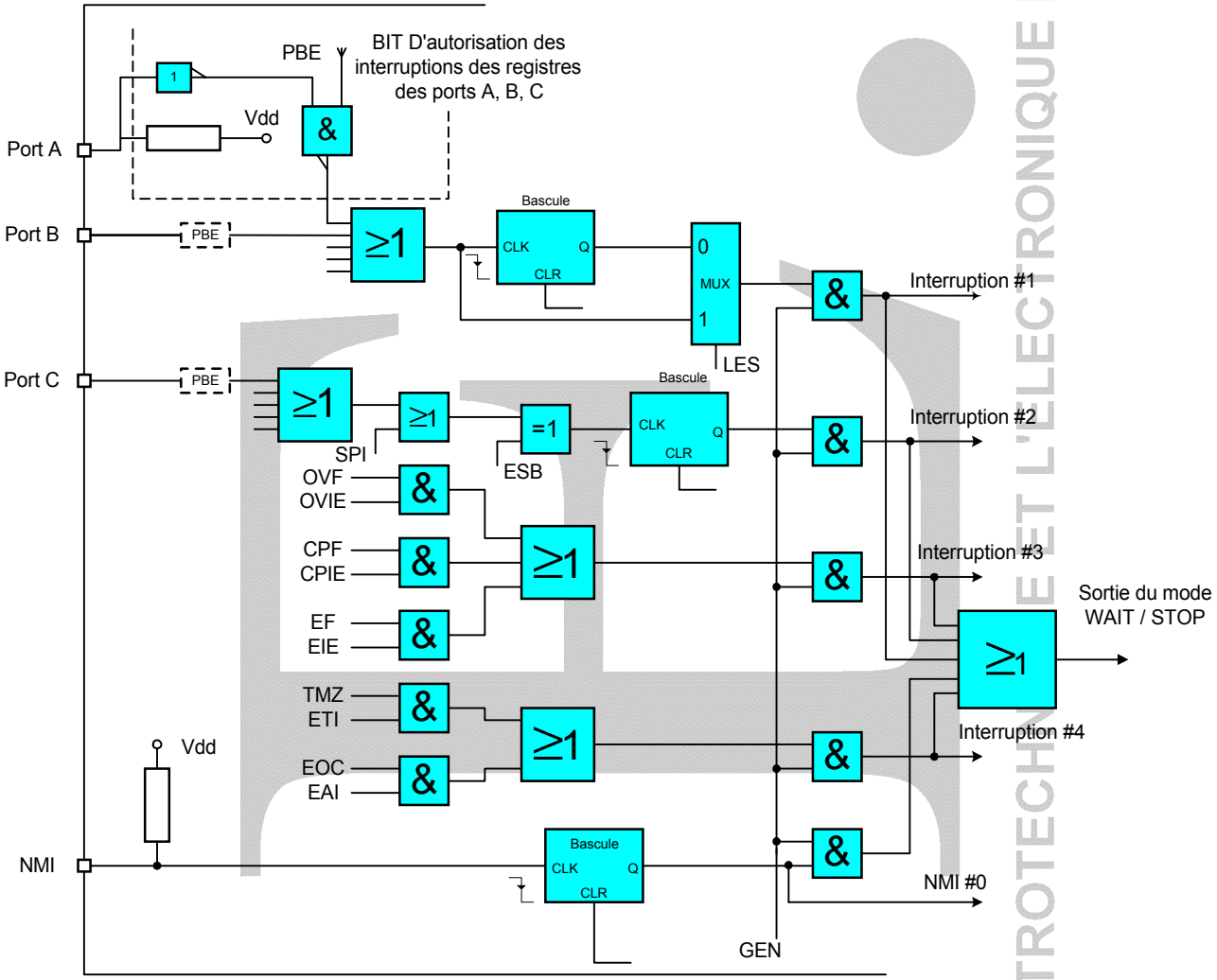
Broche NMI	**Vecteur #0**
Port A ou B	**Vecteur #1**
Port C, SPI	**Vecteur #2**
ARTimer	**Vecteur #3**
Timer et CAN	**Vecteur #4**
RESET	

L'ensemble des interruptions est géré par le registre d'interruption IOR (Interrupt Option Register)

☞ **IOR** : Registre des options des interruptions *(Interrupt Option Register).*
Adresse : C8h – Ecriture seule.

-	LES	ESB	GEN	-	-	-	-

Schéma des interruptions :



Exercice : Registre IOR.

A partir du schéma des interruptions :

- Quelles sont les interruptions que l'on peut bloquer avec le bit GEN ?

- A quoi correspond PBE ?

- A quoi sert le bit LES ?

- A quoi sert le bit ESB ?

Fin d'une interruption :

Cela se fait avec l'instruction **RETI**, il faut reprendre le cours du traitement avant l'interruption. Cela se fait par l'intermédiaire de la pile. Lors d'une interruption, l'adresse en cours est sauvegardée sur la pile (en fait le compteur programme (PC) est empilé). En fin d'interruption le contenu de la pile est dépilée vers le compteur programme. Les six registres de la pile permettent donc six interruptions simultanées (il faut aussi tenir compte des sous-programmes qui utilisent également la pile de la même façon).

Le traitement d'une interruption suit l'organigramme donné page suivante :

Exercice : organigramme des interruptions :

Dans l'organigramme :
- repasser en bleu le fonctionnement normal du CPU,
- repasser en rouge la demande d'interruption,
- repasser en vert la sortie d'interruption.

- Que se passe-t-il si le CPU reçoit une instruction RETI alors qu'il est en mode normal ?

- Que se passe-t-il si la pile est vide bien que le CPU soit en mode NMI et quand cela peut-il se produire ?

Erreur! Aucune rubrique spécifiée.

2.7 Reset

Le ST62 démarre sur un reset, en mode interruption non masquable afin que le processus d'initialisation ne puisse être interrompu. L'organigramme du reset est le suivant :

Erreur! Aucune rubrique spécifiée.

A la fin du Reset, le microcontrôleur est toujours en mode NMI, il convient donc de placer une instruction RETI afin de sortir le ST62 de ce mode. La pile étant vide, le programme s'exécutera normalement après cette instruction.

Il y a quatre sources de Reset :
- Lorsque le chien de garde atteint la valeur 0.
- Un niveau logique 0 sur l'entrée Reset du microcontrôleur.
- Lors de la mise sous tension (Power On Reset).
- Détection d'une tension d'alimentation insuffisante (Low Voltage Detector).

Les différents registres sont initialisés comme suit :
- Entrées-sorties en mode Entrée avec résistance de rappel
- Timer à rechargement automatique arrêté
- SPI et CAN désactivés.

2.8 Mise en œuvre du ST6260 :

2.8.1 Introduction :

Le ST62 a été conçu pour fonctionner dans un environnement difficile :

Erreur! Aucune rubrique spécifiée.

2.8.2 Alimentation :

Le ST62 supporte une alimentation de 3 à 6V. (V_{SS} : masse)
Pour : 3 V f_{max} = 2 MHz
5 V f_{max} = 8 MHz
6 V f_{max} = 8 MHz
Eventuellement, un condensateur de découplage de 10 µF.

2.8.3 Oscillateur :

L'oscillateur peut être obtenu :
- Par un circuit RC : de précision faible et dépendant de la température,
- Par un signal externe : nécessite un oscillateur externe,
- Par un Quartz permet une grande précision et une faible dérive en température jusqu'à 8 MHz.
- Par l'oscillateur auxiliaire

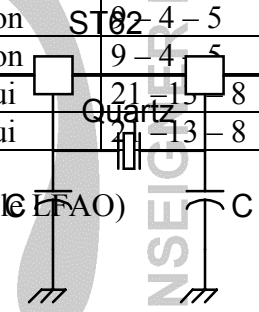
Circuit RC : Résistance entre OSC_{out} et la masse. Le condensateur est intégré au ST62.
R = Pour f = 270 kΩ pour f_{osc} = 1 MHz
R = 100 kΩ pour f = 2 MHz
R = 18 kΩ pour f_{osc} = 8 MHz

Les valeurs de fréquences sont approximatives (tolérance des résistances, dérive en température)

Oscillateur externe: Sortie de l'oscillateur connecté sur OSC_{in}

Quartz : un quartz entre OSCin et OSCout ainsi que deux capacités de 15 pF à 22 pF entre OSCin et la masse et OSCout et la masse. L'ensemble doit être placé au plus prés du circuit

On peut également utiliser un résonateur.



2.8.4 Reset

Pour initialiser le microcontrôleur on utilise l'entrée RESET active à l'état bas.
Un simple circuit RC de constante de temps 10ms et un interrupteur aux bornes du condensateur permet de réinitialiser le microcontrôleur.

2.9 La famille ST625X et 626X

2.9.1 Désignation :

La désignation des microcontrôleurs suit la schéma suivant :

ST62 E 52 B B 1
T 53 C M 6
P 55 N 3
60
62
63
65

ST62 désigne la famille ST62
Les lettres E : version EEPROM
T : Version OTP (programmable une seule fois)
P : Version FASTROM (programmée par ST)

Le type de microcontrôleur :

	EPROM	EEPROM	Boîtier	SPI	E/S – An- 20mA
53	1836		DIP20	Oui	13 – 7 – 6
60	3884	128	DIP20	Oui	13 – 7 – 6
63	1836	64	DIP20	Oui	13 – 7 – 6
52	1836	-	DIP16	Non	9 – 4 – 5
62	1836	64	DIP16	Non	9 – 4 – 5
55	3884		DIP28	Oui	21 – 15 – 8
65	3884	128	DIP28	Oui	21 – 13 – 8

Version : B ou C (les versions C possèdent Le LVD, L'OSG ainsi que le EFAO)

Le Boîtier : B → DIP ; M → SO ; N → SOP

Le suffixe 1 , 3 ou 6 (c.f. caractéristiques électriques)

Le kit ST626X permet la programmation des microcontrôleurs ST625X et ST626X.

Ainsi, un ST62E60CB6
Est la version C, effaçable aux UV, en boîtier DIP28, suffixe 6 du ST6260.

2.9.2 Caractéristiques électriques

Caractéristiques maximum :

Symbole	Paramètres	Conditions de Test	Unité
V_{DD}	Alimentation	- 0,3 à 7,0	V
V_I	Tension d'entrée	V_{SS} – 0,3 à V_{DD} + 0,3 (1)	V
V_O	Tension de sortie	V_{SS} – 0,3 à V_{DD} + 0,3 (1)	V
IV_{DD}	Courant max dans V_{DD}	80	mA
IV_{SS}	Courant max dans V_{SS}	100	mA
T_j	Température de Jonction	150	°C
T_{STG}	Température de stockage	-60 à 15	°C

(1) Dans ces limites, les diodes de protection ne conduisent pas. Les tensions en dehors de ces limites sont autorisées aussi longtemps que les courants d'entrées restent dans les spécifications.

Spécifications recommandées :

Symbole	Paramètres	Conditions de Test	Valeurs			Unité
			Min.	Typ.	Max.	
T_A	Température de fonctionnement	Version suffixe 6 Version suffixe 1 Version suffixe 3	-40 0 -40		85 70 125	°C
V_{DD}	Tension de fonctionnement Version C	f_{OSC} = 4 MHz suffixes 1 &6 f_{OSC} = 4 MHz suffixe 3 f_{OSC} = 8 MHz suffixes 1 &6 f_{OSC} = 8 MHz suffixe 3	3,0 3,0 3,6 4,5		6,0 6,0 6,0 6,0	V
	Tension de fonctionnement Version B	f_{OSC} = 4 MHz suffixes 1 &6 f_{OSC} = 4 MHz suffixe 3 f_{OSC} = 8 MHz suffixes 1 &6 f_{OSC} = 8 MHz suffixe 3	3,0 3,0 4,0 4,5		6,0 6,0 6,0 6,0	V
f_{OSC}	Fréquence de l'oscillateur Version C	V_{DD} = 3,0 V suffixes 1 &6 V_{DD} = 3,0 V suffixe 3 V_{DD} = 3,6 V suffixes 1 &6 V_{DD} = 3,6 V suffixe3	0 0 0 0		4,0 4,0 8,0 4,0	MHz
	Fréquence de l'oscillateur Version C	V_{DD} = 3,0 V suffixes 1 &6 V_{DD} = 3,0 V suffixe 3 V_{DD} = 3,6 V suffixes 1 &6 V_{DD} = 3,6 V suffixe3	0 0 0 0		4,0 4,0 8,0 4,0	
I_{INJ+}	Courant d'entrée positif	V_{DD} = 4,5 à 5,5 V			+ 5	mA
I_{INJ-}	Courant d'entrée négatif	V_{DD} = 4,5 à 5,5 V			- 5	mA

Symbole	Paramètres	Conditions de Test	Valeurs			Unité
			Min.	Typ.	Max.	
V_{IL}	Tension d'entrée pour un 0				V_{DD}×0,3	V
V_{IH}	Tension d'entrée pour un 1		V_{DD}×0,7			V
V_{Hys}	Tension d'hystérésis		0,2			V
V_{up}	LVD : détection de mise sous tension			4,1	4,3	V
V_{dn}	LVD: détection de mise hors tension		3,5	3,8		V
V_{OL}	Tension de sortie pour un 0	V_{DD} = 5V ; I_{OL} = 10 µA V_{DD} = 5V ; I_{OL} = 3 mA V_{DD} = 5V ; I_{OL} = 10mA			0,1 0,8 1,2	V
	Tension de sortie pour un 0 Sur les entrées 30 mA	V_{DD} = 5,0 V ; I_{OL} = 10 µA V_{DD} = 5,0 V ; I_{OL} = 7 mA V_{DD} = 5,0 V ; I_{OL} = 15 mA V_{DD} = 5,0 V ; I_{OL} = 30 mA			0,1 0,8 1,3 2,0	V
V_{OH}	Tension de sortie pour un 1	V_{DD} = 5,0 V ; I_{OH} = - 10 µA V_{DD} = 5,0 V ; I_{OH} = - 5 mA	4,9 3,5			V
R_{PU}	Résistance de tirage	Toutes les entrées	40	100	350	kΩ
		Reset	150	350	900	
Cin ; Cout	Capacité des E/S				10	pF
I_{IL}	Courant d'entrée pour un 0	V_{IN} = V_{SS}		0,1	1	µA
I_{IH}	Courant d'entrée pour un 1	V_{IN} = V_{SS}	-8	-16	-30 10	µA
I_{DD}	Courrant d'alimentation en mode RESET				7	mA
	Courrant d'alimentation en mode RUN				7	mA
	Courrant d'alimentation en mode WAIT				2,5	mA
	Courrant d'alimentation en mode STOP LVD désactivé				20	µA
	Courrant d'alimentation en mode RESET LVD activé				500	nA
V_{up}	Détection de mise sous tension LVD		V_{dn} + 50 mV	4,1	4,3	V
V_{DN}	Détection de mise hors tension		3,6	3,8	V_{up} – 50 mV	V

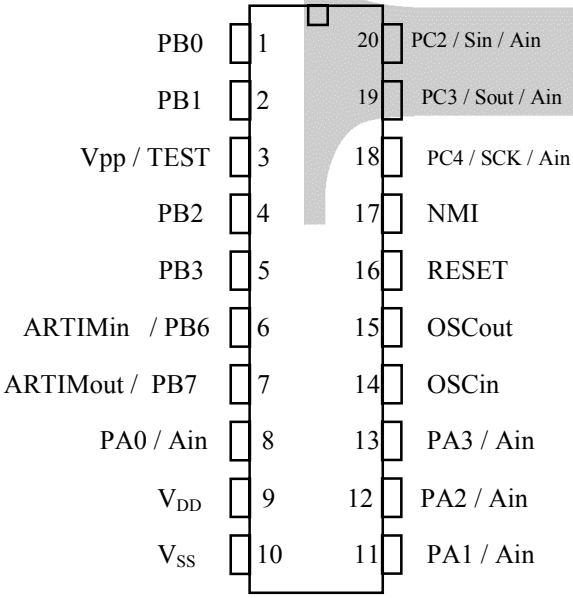
Caractéristiques de l'oscillateur :

Symbole	Paramètres	Conditions de Test	Valeur			Unité
			Min.	Typ.	Max.	
f_{LFAO}	Fréquence interne LFA activé		200	400	800	kHz
f_{OSG}	Fréquence interne OSG activé	V_{DD} = 3 V V_{DD} = 3,6 V V_{DD} = 4,5 V	2 2 4		F_{OSC}	MHz
f_{RC}		V_{DD} = 5,0 V Version C R = 47 kΩ R = 100 kΩ R = 470 kΩ	 4 2,7 800	 5 3,2 850	 5,8 3,5 900	 MHz MHz kHz
		V_{DD} = 5,0 V Version B R = 10 kΩ R = 27 kΩ R = 100 kΩ	 2,4 1,8 800	 3,1 2,2 980	 3,8 2,5 1200	 MHz MHz kHz

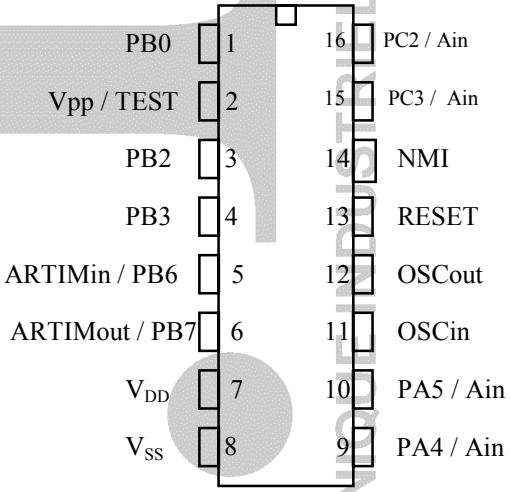
Caractéristiques de l'EEPROM :

Symbole	Paramètres	Conditions de Test	Valeur			Unité
			Min.	Typ.	Max.	
Rétention	Temps de rétention de l'EEPROM	T_A = 55 °C	10			Ans
T_{WEE}	Temps d'écriture	TA = 25°C TA = 85°C TA = 125°C		5 10 20	10 20 30	ms
Réécritures			$3 .10^5$	10^6		cycles

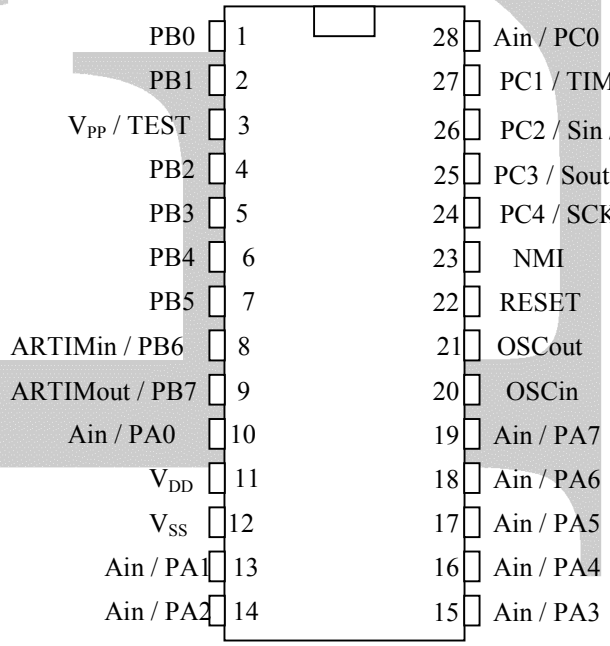
2.9.1 Brochages :



ST62X53 / 62X60 / 62X63



ST62X52 / 62X62



ST62X55 / 62X65

ST62X65 /62X55

3 STRUCTURE LOGICIELLE DES SYSTEMES DE TRAITEMENT DE L'INFORMATION

3.1 Le Programme :

Un programme utilisateur est une réalisation logicielle qui est implantée dans la mémoire d'un système de traitement. Il définit, à l'aide d'instructions, les opérations que doit réaliser le processeur sur les données d'entrée pour obtenir les données de sortie.
Les instructions sont exécutées les unes après les autres, dans l'ordre de leur implantation en mémoire.

3.2 Les Langages :

■ Langage machine

Le langage machine est le langage compréhensible par le microprocesseur au niveau du décodeur d'instruction. Il est constitué uniquement de nombres binaires.

■ Langage assembleur

Il est pratiquement impossible de lire un programme en langage machine codé en binaire. Il est encore très fastidieux de lire un programme en langage machine codé en hexadécimal.
Pour faciliter le travail de programmation, les concepteurs ont développé pour chaque système à microprocesseur, un logiciel appelé assembleur qui traduit les codes mnémoniques en langage machine.
Le langage de programmation d'un microprocesseur à base de mnémonique est appelé langage assembleur. Le programme originel en langage assembleur est le programme source. Le programme traduit en langage machine est le programme objet.
La programmation en assembleur est une programmation au plus prés du processeur. C'est un travail fastidieux qui demande de connaître parfaitement la structure matérielle du système utilisé.

Ce langage est utilisé dans les cas suivants :

- Programme simple
- Taille du programme critique
- Recherche d'un code très rapide

■ Langages évolués (BASIC, PASCAL, Langage C,...)

Les langages évolués ne dépendent pas du processeur utilisé. Deux types de stratégies sont envisagées :

Langage interprété : Les instructions du langage évolué sont traduites à la volée par l'interpréteur qui traduit ces instructions en langage machine. Si une instruction est exécutée plusieurs fois elle sera traduite plusieurs fois ce qui ralentit l'exécution générale du programme.

Langage compilé : Avant l'exécution du programme, le compilateur traduit une fois pour toute l'ensemble des instructions évoluées en langage machine. Le code est plus rapide mais la mise au point plus longue (phase de compilation)

Chaque langage possède son domaine de prédilection (Base de donnée, calcul scientifique etc...)

3.3 Les instructions

3.3.1 Structure générale d'une instruction

Un programme est une suite ordonnée d'instructions élémentaires que le processeur doit exécuter séquentiellement pour élaborer des commandes.

Une instruction est constituée de deux champs :
- L'opération qui définit le type de travail à effectuer ;
- L'opérande qui définit l'objet sur lequel doit être effectué le travail.

Un commentaire associé à chaque ligne d'instruction est très utile, voire indispensable, pour le programmeur ou pour le technicien de maintenance.
Le numéro d'ordre d'instruction permet de visualiser facilement l'ordonnancement des instructions.

3.3.2 Modes d'adressage

Les modes d'adressage caractérisent la façon dont les opérandes sont échangés entre les périphériques, la mémoire et les registres de l'unité de traitement.

Adressage Immédiat
Le code opération de l'instruction est suivi immédiatement d'une valeur numérique qui est l'opérande.
On note dans l'organigramme : nn -----> R
On lit : transférer la valeur immédiate nn dans le registre R.
Exemple : charger la valeur décimale 19 dans l'accumulateur A.
On note : 19 ------> A
En langage assembleur «ST62 » l'instruction qui charge une valeur immédiate (constante donnée par le programme) dans l'accumulateur A est : LDI A,19.
En assembleur, les instructions équivalentes à LDI A,19 sont :

```
LDI A,13h
LDI A,00010011b
```

Le symbole « h » indique que la donnée immédiate est en hexadécimale ($13_h=19_{10}$).
Le symbole « b » indique que la donnée suivante est en binaire ($00010011_2 = 19_{10}$).
Si rien n'est précisé cette donnée est décimale.

Dans tous les cas, en langage machine, cette instruction s'écrit :
En hexadécimal 17 13 ;
En binaire 0001 0111 0001 0011.
On constate que l'instruction de chargement en adressage immédiat nécessite 2 octets.

Adressage direct
Le code opération de l'instruction est suivi d'une adresse mémoire. La donnée contenue dans l'adresse mémoire est l'opérande de cette opération.
On note dans l'organigramme : (e) ------> R.
On lit : transférer le contenu de la mémoire repérée par l'adresse e dans le registre R.
Exemple : charger dans l'accumulateur A le contenu de l'adresse 08Fh.
On note (08Fh) -----> A
En langage assembleur « ST62 » l'instruction qui charge la valeur contenue à l'adresse 08fh de la mémoire dans l'accumulateur A est LD A,08Fh

LD est la mnémonique de l'instruction « charger » (LOAD A).
Le symbole « h » signifie que l'adresse est donnée en hexadécimal.
En langage machine, cette instruction s'écrit :

En hexadécimal 1Fh 8Fh ;
En binaire 0001 1111 1000 1111.

1Fh est le code opération du chargement dans l'accumulateur d'une donnée en adressage étendu.
« 8F » est l'octet de l'adresse contenant la donnée. On constate que l'instruction de chargement dans l'accumulateur A en adressage étendu nécessite 2 octets.

Adressage indirect

L'adresse de l'opérande est donnée par le registre d'indirection X ou Y
Exemple : LDI X, 038h
LD A, (X)

Est équivalent à : LD A, 038h

Adressage implicite (inhérent) ou adressage registre

Le code opération contient implicitement l'opérande de l'opération. Il ne faut donc pas préciser une adresse ou une donnée après le code opération.
Ce type d'adressage est utilisé pour un transfert de registre à registre, pour initialiser un registre, ou pour faire une opération sur un registre (complémentation, incrémentation, décalage, rotation,...)
Exemple mettre la valeur « 0 » dans l'accumulateur s'écrit CLR A en assembleur (CLEAR A).
En langage machine le code opération de cette instruction s'écrit :

En hexadécimal DFh FFh ;
En binaire 1101 1111 1111 1111.

Adressage immédiat

Utilisé par les instructions de saut JP et CALL
L'adresse du saut est donnée de manière exacte sur 12 bits

Adressage relatif

Utilisé par les instructions de saut conditionnel, donne ici le saut à effectuer à partir de l'adresse en cours (décalage positif ou négatif)

Le nombre de mode d'adressage et leur complexité fait partie des éléments permettant de déterminer la puissance du microprocesseur.

TABLEAU DES PRINCIPALES INSTRUCTIONS ET ADRESSAGES DU MICROCONTROLEUR ST62

Instructions du microcontrôleur ST62	Mnémonique	Cycles	Adressage	Flag
Instructions de mouvement en mémoire				
Placer dans l'accumulateur A l'accumulateur R	LD A,R	4	DIRECT	Z
Placer dans l'accumulateur R l'accumulateur A	LD R,A	4	DIRECT	Z
Placer dans l'accumulateur A le registre pointé par X	LD A,(X)	4	INDIRECT	Z
Placer dans le registre pointé par X l'accumulateur A	LD (X),A	4	INDIRECT	Z
Placer dans l'accumulateur A le registre pointé par X	LD A,(Y)	4	INDIRECT	Z
Placer dans le registre pointé par X l'accumulateur A	LD (Y),A	4	INDIRECT	Z
Placer dans A le registre rr	LD A,rr	4	DIRECT	Z
Placer dans le registre rr l'accumulateur A	LD rr,A	4	Direct	Z
Placer nn dans le registre R	LDI R,nn	4	Immédiat	
Instructions arithmétiques et logiques				
Additionner A avec le registre rr	ADD A, rr	4	DIRECT	Z, C
Additionner A avec le registre pointé par X	ADD A, (X)	4	Indirect	Z, C
Additionner A avec le registre pointé par Y	ADD A, (Y)	4	Indirect	Z, C
Additionner A avec la donnée nn	ADDI A, nn	4	Immédiat	Z, C
Faire A ET le registre rr (A ← A.rr)	AND A, rr	4	DIRECT	Z, C
Faire A ET le registre pointé par X (A ← A.(X))	AND A, (X)	4	Indirect	Z, C
Faire A ET le registre pointé par Y (A ← A.(Y))	AND A, (Y)	4	Indirect	Z, C
Faire A ET la donnée nn (A ←A.rr)	ANDI A, nn	4	Immédiat	Z, C
Effacer A	CLR A	4	Inhérent	Z, C
Effacer le registre rr	CLR rr	4	Direct	
Faire le complément de A dans A	COM A	4	Inhérent	Z, C
Comparer A et R	CP A,R	4	Direct	Z, C
Comparer A et le registre pointé par X	CP A,(X)	4	Indirect	Z, C
Comparer A et le registre pointé par Y	CP A,(Y)	4	Indirect	Z, C
Comparer A et le registre rr	CP A,rr	4	Direct	Z, C
Comparer A et nn	CPI A,nn	4	Immédiat	Z, C
Décrémenter le registre R	DEC R	4	Direct	Z
Décrémenter le registre pointé par X	DEC (X)	4	Indirect	Z
Décrémenter le registre pointé par Y	DEC (Y)	4	Indirect	Z
Décrémenter le registre rr	DEC rr	4	Direct	Z
Incrémenter le registre R	INC R	4	Direct	Z
Incrémenté le registre pointé par X	INC (X)	4	Indirect	Z
Incrémenté le registre pointé par Y	INC(Y)	4	Indirect	Z
Incrémenter le registre rr	INC rr	4	Direct	Z
Rotation vers la gauche du registre A	RLC A	4	Inhérent	Z, C
Décalage vers la gauche du registre A	SLA A	4	Inhérent	Z, C
Soustraire A à R dans A	SUB A,R	4	Direct	Z, C
Soustraire A au registre pointé par X dans A	SUB A,(X)	4	Indirect	Z, C
Soustraire A au registre pointé par Y dans A	SUB A,(Y)	4	Indirect	Z, C
Soustraire A à au registre rr dans A	SUB A, rr	4	Direct	Z, C
Soustraire A à nn dans A	SUBI A,nn	4	Indirect	Z,C

Instructions du microcontrôleur ST62	Mnémonique	Cycles	Adressage	Flag
Instruction de saut conditionnel				
Aller à e si C = 0	JRC e	2	Relatif ± 15	
Aller à e si C = 1	JRNC e	2	Relatif ±15	
Aller à e si Z = 1	JRZ e	2	Relatif ±15	
Aller à e si Z = 0	JRNZ e	2	Relatif ± 15	
Aller à e si le bit n°b registre rr = 0	JRR b,rr,ee	5	Relatif ± 128	C
Aller à e si le bit n°b registre rr = 1	JRS b,rr,ee	5	Relatif ± 128	C
Instructions de saut inconditionnel				
Aller à abc	JP abc	4	Etendu	
Aller au sous programme en abc	CALL abc	4	Etendu	
Fin du sous programme	RET	2	Inhérent	
Fin de l'interruption	RETI	2	Inhérent	Z, C
Instructions sur bit				
Forcer à 0 le bit n° b de A	RES b,A	4	Direct	
Forcer à 0 le bit n° b du registre rr	RES b,rr	4	Direct	
Forcer à 1 le bit n° b de A	SET b,A	4	Direct	
Forcer à 1 le bit n° b du registre rr	SET b,rr	4	Direct	
Instructions diverses				
Pas d'opération	NOP	4	Inhérent	
Arrêt du processeur	STOP	4	Inhérent	
Processeur en attente	WAIT	4	Inhérent	

R correspond au registre A, X, Y, V ou W
rr correspond à l'adresse d'un registre en mémoire de donnée sur 8 bits
nn est une donnée sur 8 bits
e est une adresse de la mémoire de programme sur 12 bits
b correspond à un numéro de bit de 0 à 7 :

7	6	5	4	3	2	1	0

Exercice : ALU du ST62.

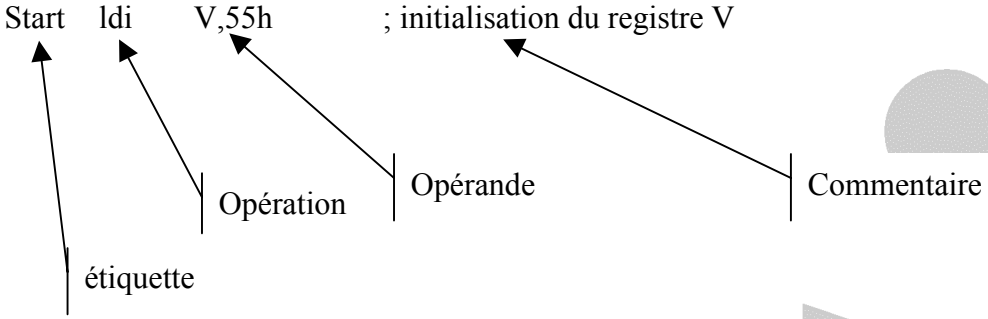
Faire l'inventaire des opérations effectuées par l'ALU du ST62.

3.4 Assembleur du ST62

L'assembleur du ST62 est un programme : AST6. Il est commun à toute la famille des ST62

Le code source est un fichier ASCII créé avec n'importe quel éditeur de texte (EDIT ou NOTEPAD).

Chaque ligne de code possède jusqu'à quatre champs :



L'étiquette :
Le premier champ commence dés le début de la ligne (pas d'espace, sinon l'assembleur considère qu'il s'agit d'une opération).
Il s'agit d'une étiquette afin de référencer certain point du programme pour les instructions de saut.
Ce champ est facultatif.

L'opération :
Un espace au minimum doit séparer l'étiquette de l'opération. S'il n'y a pas d'étiquette, il faut insérer un espace en début de ligne.
Il s'agit du mnémonique de l'opération du microcontrôleur.

L'opérande :
Un espace au minimum doit séparer l'opérande de l'opération.
Il peut y avoir deux opérandes séparés d'une virgule.
L'opérande peut ne pas exister (adressage inhérent).

Commentaire :
Un commentaire commence par un point virgule.
Il est très important de commenter un programme afin qu'il puisse être repris par quelqu'un d'autre que le concepteur du programme ou tout simplement pour pouvoir le reprendre plus tard.

Les directives :

Une directive commence par un point.
Ce sont des instructions destinées à l'assembleur, elles sont destinées à aider le programmeur.

Les principales directives sont les suivantes :

.VERS

Indique le type de microcontrôleur utilisé.
Ex :
.Vers "ST6260"

.ROMSIZE
Indique la taille en ko de la ROM du microcontrôleur.
Ex :
.ROMSIZE 4

.INPUT

Permet d'insérer un fichier ici le fichier texte. Cette directive est très utile, en effet un programme commence souvent de la même façon ou alors on retrouve souvent des bouts de codes identiques d'une application à une autre. Il suffit d'écrire ce code dans un fichier et d'ajouter la directive .INPUT associée au nom du fichier.
Ex:
.INPUT "6260_REG.ASM"
Ici, il s'agit des déclarations de registre (noms et adresse de tous les registres).

.ORG
Définit l'emplacement en mémoire de programme à partir duquel le code va être implanté.

Exemple :

```
.ORG 0ffeh
        JP Debut
```

Cette directive suivit de l'instruction JP définit le vecteur Reset

3.5 Mise en œuvre logicielle du ST6260

Le ST6260 démarre sur le vecteur RESET en mode NMI. Il faut donc penser à donner au vecteur RESET l'adresse de début de programme et terminer la routine d'initialisation par un RETI afin de sortir le processeur du mode NMI

```
        .VERS "ST6260"          ; le microcontrôleur est un ST6260
        .ROMSIZE 4              ; La mémoire programme fait 4 ko
        .INPUT "6260_REG.ASM"   ; insertion des déclarations de registre du ST6260

                        ; Initialisation des vecteurs d'interruption
        .ORG  0ffeh
        JP    debut
        :
        :
        :
        . ORG 0880h  ; zone utilisateur en mémoire de donnée
debut   :            ; Initialisation du ST62
        :            ; Ports d'entrées sorties, interruptions etc.
        :
        RETI
```

Ne pas oublier le chien de garde, s'il est activé, en insérant régulièrement des instructions LDI wdr, 0ffh.

3.6 Exemples de programmes

3.6.1 Utilisation du CAN sans interruption

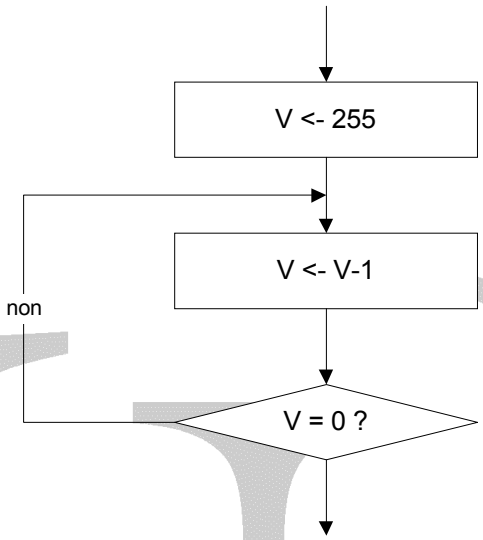
Exercice : CAN sans interruption.

Reprendre l'algorithme de l'utilisation du CAN sans interruption et écrire le programme.

3.6.2 Temporisation

Exercice : Temporisation.

Le langage machine est parfois trop rapide, on est alors obligé d'insérer une boucle de retard permettant de ralentir le programme.
L'algorithme est le suivant :



Ecrire le programme correspondant à cette boucle

Exprimer le temps de cette boucle en fonction de la valeur initiale du registre V. En déduire les temps mini et maxi de ce retard. Le ST6260 est cadencé à 8 MHz.

On peut avoir besoin de retards plus importants, dans ce cas, il faut imbriquer deux boucles.
Ecrire l'algorithme et le programme de ces deux boucles imbriquées.

Le Site
E.I.
ENSEIGNER L'ELECTROTECHNIQUE ET L'ELECTRONIQUE INDUSTRIELLE

Exercice : CAN avec interruption.

Ecrire le programme utilisant le CAN avec interruption

3.7 Conclusion :

Plusieurs points n'ont pas été abordés dans ce cours:

- Gestion des modes économie d'énergie (Instructions WAIT et STOP)
- Gestion des interruptions
- Programmation graphique (ST REALIZER étudiée en TP + Documentation)
- Programmation des microcontrôleurs (étudiée en TP)
- Starter Kit (étudié en TP)
- Fonctions évoluées de AST6 et LST6 (étudiées ponctuellement en TP)

Pour compléter les informations données dans ce cours, il est nécessaire de se reporter à la documentation fournie par ST MICROELECTRONICS.

Bibliographie :

- LE MANUEL DU MICROCONTROLEUR ST62 (ELEKTOR)
- DATASHEETS ST des différents produits
- WWW.ST.COM