NF 15
Semestre 2008

1ère partie

P. Pouletaut
J.-F. Lerallut

MICROPROCESSEURS
INTERFACES ET LOGICIELS DE BASE
NOTES DE COURS

Université de Technologie de Compiègne

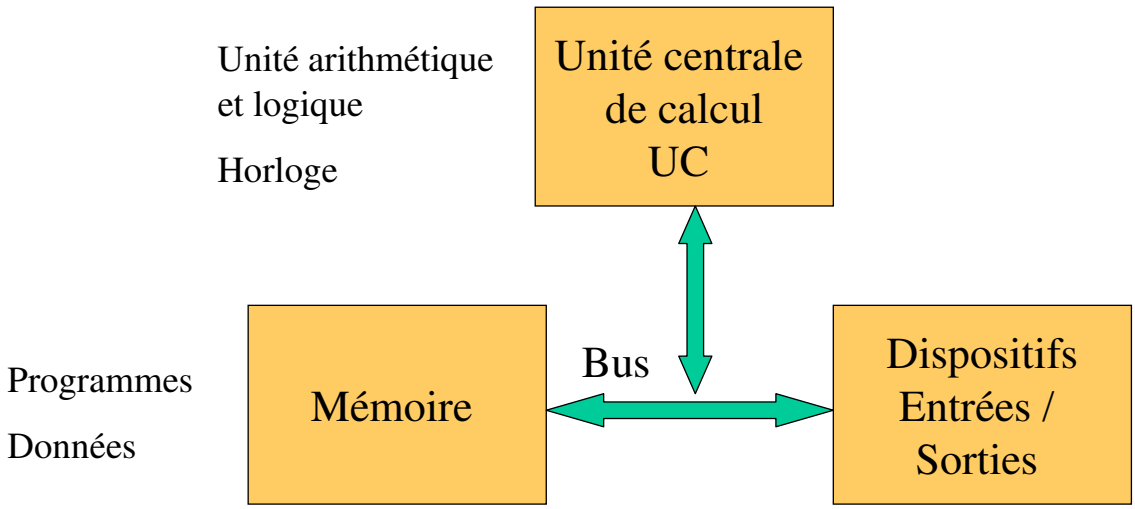
Cours NF15

Chapitre 1 : Introduction aux microcontrôleurs

- Sommaire

1.1 Architecture des ordinateurs

- Composants d'un ordinateur

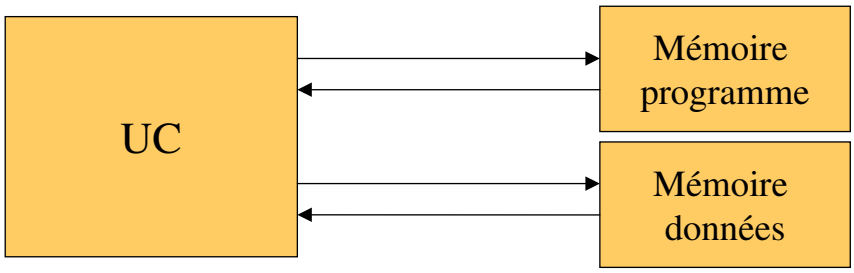


1.1 Architecture des ordinateurs

• Organisation de la mémoire

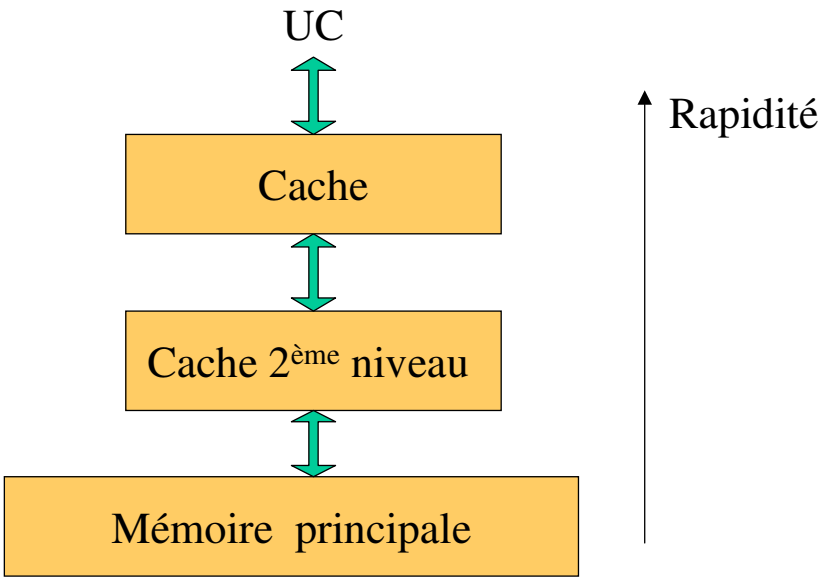
zones de programmes (instructions) et
zones de données (valeurs numériques)

- Architecture Von Neumann : instructions et données dans une seule mémoire
  architecture dépréciée, lent
- Architecture Harvard : instructions et données dans deux mémoires séparées
  architecture très utilisée car plus rapide (l'UC a accès simultanément à l'instruction et aux données associées), structure interne plus complexe



1.1 Architecture des ordinateurs

• Hiérarchie des mémoires



1.1 Architecture des ordinateurs

- Types de logiciel
  - Noyau :

    exécuteur de programme, cœur du système exécutif

  - Système d'exploitation :

    gestionnaire de la mémoire et du programme d'exécution

    (ex. : IRIX, MacOS, Windows 2000, Unix, Linux)

  - Programme d'application :

    traitement de texte, tableur, lecteur PDF, ...

    (ex. : OpenOffice, Word, Gimp, Adobe Acrobat, ImageJ)

1.2 Types d'ordinateurs

- Microprocesseur
  - Ensemble d'UCs réunis dans une seule puce

    Pentium, AMD, Athlon

- Microordinateur
  - Système avec pour UC un microprocesseur

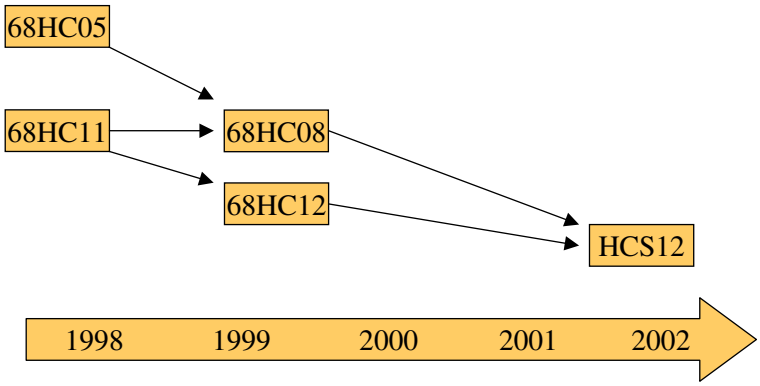
    PC, station de travail

- Microcontrôleur
  - Système avec tous les composants d'ordinateur réunis dans une puce (VLSI)
    Very Large Scale Integration (+ 100 000 transistors)

    Motorola 68HC12, Microchip PIC 16F628, Atmel AVR,
    Parallax BASIC STAMP, ST6 de STMicroelectronics

1.3 Présentation du 68HC12

- Technologie HCS12 : Motorola, micro 16 bits, cellule 0,25µm

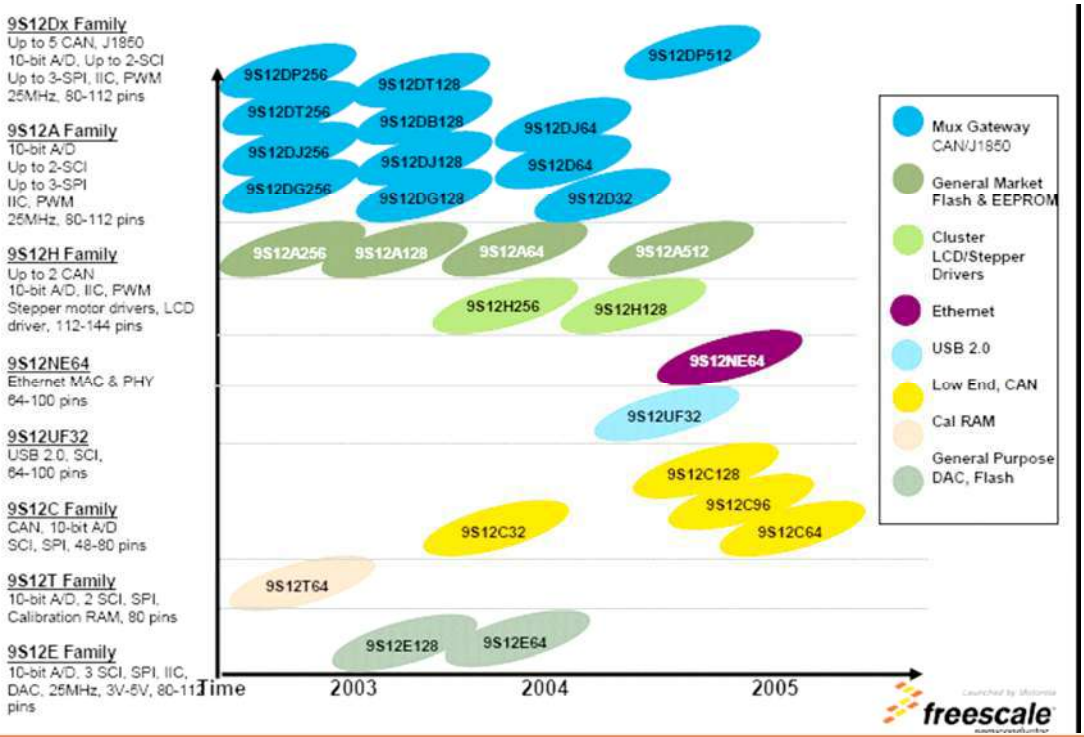


Convergence des technologies 8-bits vers 16-bits :

- faible puissance
- vitesse rapide
- langage haut niveau

1.3 Présentation du 68HC12

- Technologie HCS12 : familles



1.3 Présentation du 68HC12

- Matériel : micro 16 bits MC9S12DP256B

* 12 Ko mémoire On-Chip RAM
* 256 Ko mémoire On-Chip FLASH EEPROM
* 4 Ko mémoire On-Chip EEPROM
* 89 canaux numériques E/S (Ports A, B, K et E)
* 20 lignes numériques E/S avec interruption et capacité de veille
* Bus multiplexé externe (données 16 bits)
* Adressage indexé renforcé, capacité de mémoire étendue
* 2 ports de communication série asynchrone (SCI)
* 3 ports de communication série synchrone pour périphériques (SPI)
* 2 convertisseurs analogiques numériques 10-Bit pour 8 canaux
* Accumulateur 16 bits pour 8 canaux pulsés (PWM)
* Circuit d ’interruption temps réel
* Temporisateur 16 bits avec 8 canaux de capture ou de comparaison

1.3 Présentation du 68HC12

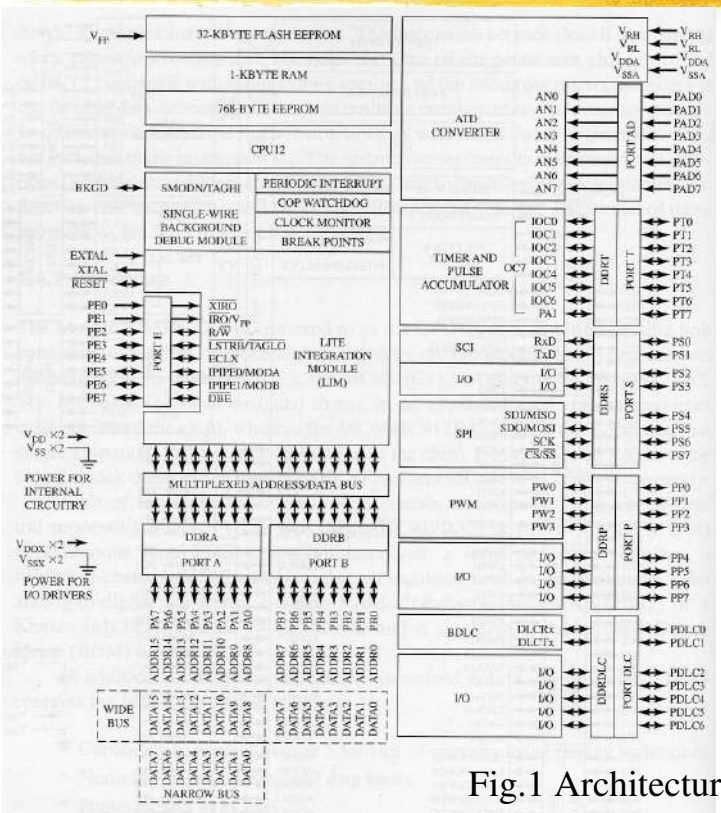


Fig.1 Architecture matérielle du 68HC12

1.3 Présentation du 68HC12

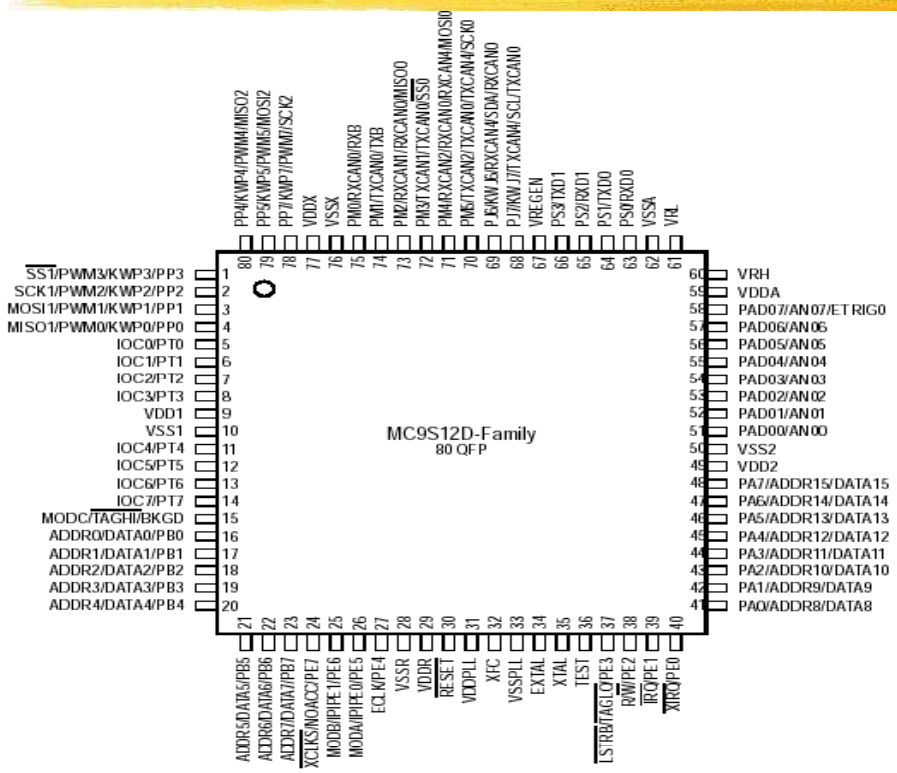


Fig.2 Brochage du MC9S12DP256B

1.3 Présentation du 68HC12

- Logiciel

  * Jeu de 209 instructions
  * Instructions du microcontrôleur parent 68HC11
  * Transfert de données
  * Opérations logiques et arithmétiques
  * Appel de sous-programmes
  * Logique floue
  * Test et branchement
  * Instructions pour données 8 ou 16 bits

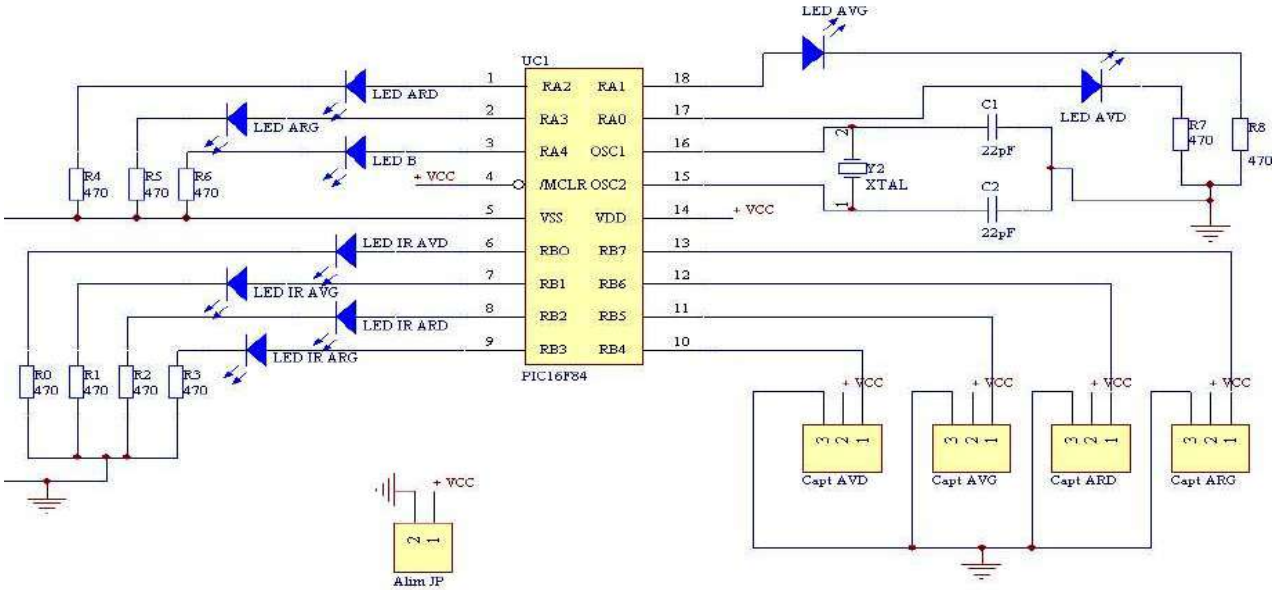
1.4 Applications

- Marché des ventes
  - ➢ Nombre de microcontrôleurs vendus bien supérieur à celui des micro-ordinateurs
    Arizona Microchip déclare avoir vendu plus de 1,5 milliards d'unités en 2001

- Applications multiples de l'électronique embarquée
  Voiture (ordinateur de bord), électroménager (programmateur), hi-fi (orgue numérique), lecteur MP3, téléphone portable,
  moniteurs cardiaques, ECG, écho-Doppler, pacemakers, capteurs intelligents mesureurs biomédicaux (pression sanguine, taux de sucre)

1.4 Applications

- Détecteur d'obstacles



1.4 Applications

- Quelques liens
  - Livres

Embedded systems design and applications with the 68HC12 and HCS12, Steven F. Barrett, Daniel J. Pack, Pearson/Prentice Hall, 2004, TK7895.E42 BAR

Introduction to Microcontrollers: Architecture, Programming, and Interfacing for the Motorola 68HC12, G. Jack Lipovski, Ed. Hardcover

68HC12 Microcontroller, Daniel J. Pack, Steven F. Barrett, Ed. Hardcover

  - Internet

http://microcontrollerssolutions.info/

http://www.microchip.com

http://www.freescale.com/

1.5 Tests

1. Quels sont les composants principaux d'un ordinateur ?

2. Qu'appelle-t-on un ordinateur, un microcontrôleur ou un microprocesseur ?

3. Quelles sont les caractéristiques principales du 68HC12 ?

4. Quel type d'architecture est aujourd'hui le plus utilisé pour les microcontrôleurs ?

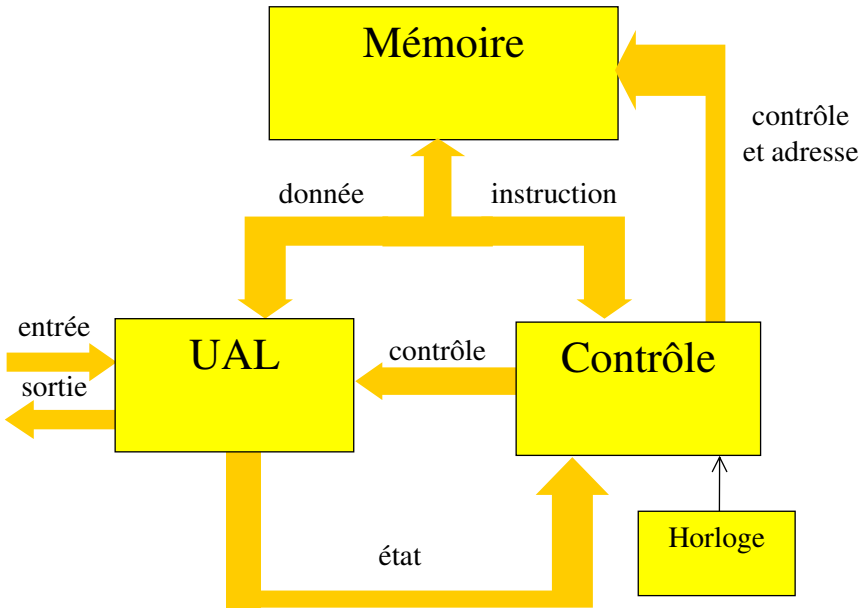
5. Donnez une définition d'un cache.

6. Hiérarchisez les mémoires selon leur rapidité.

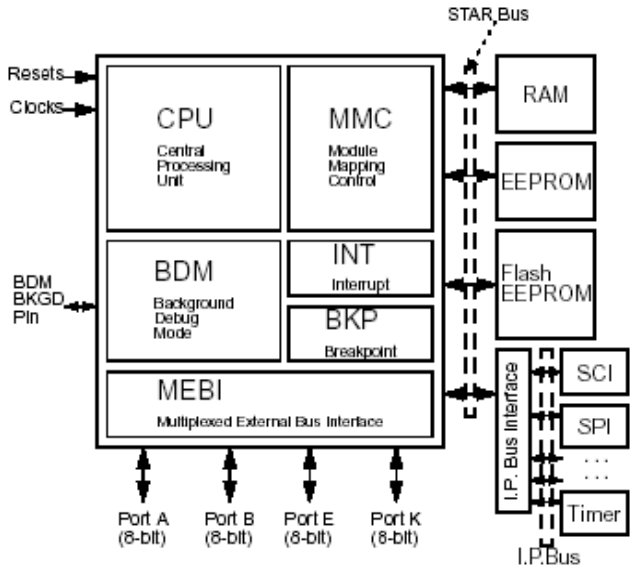
Chapitre 2 : Configuration matérielle du microcontrôleur

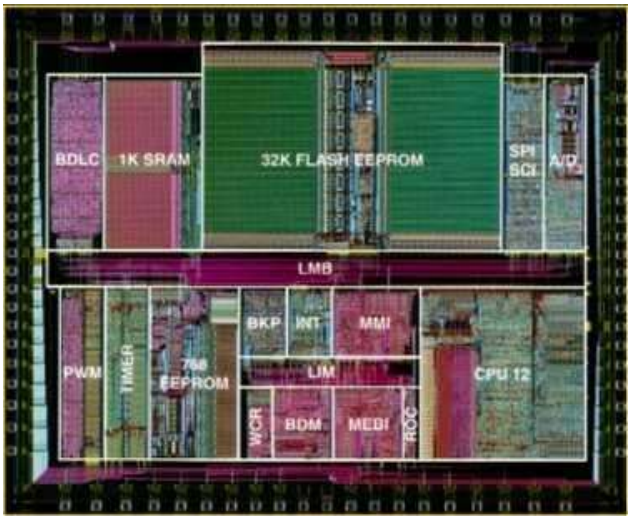
- Sommaire

2.1 Architecture matérielle

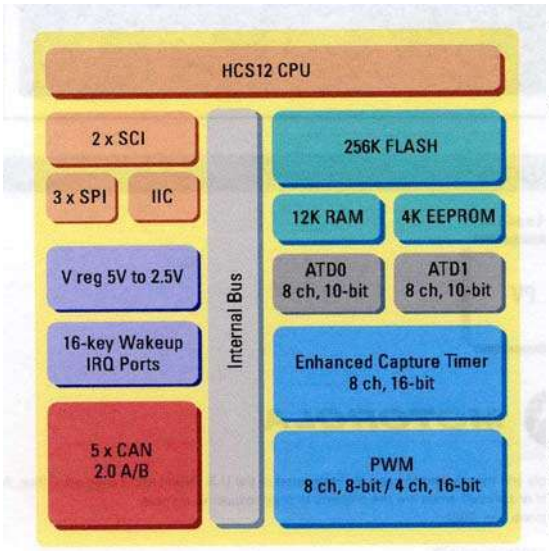


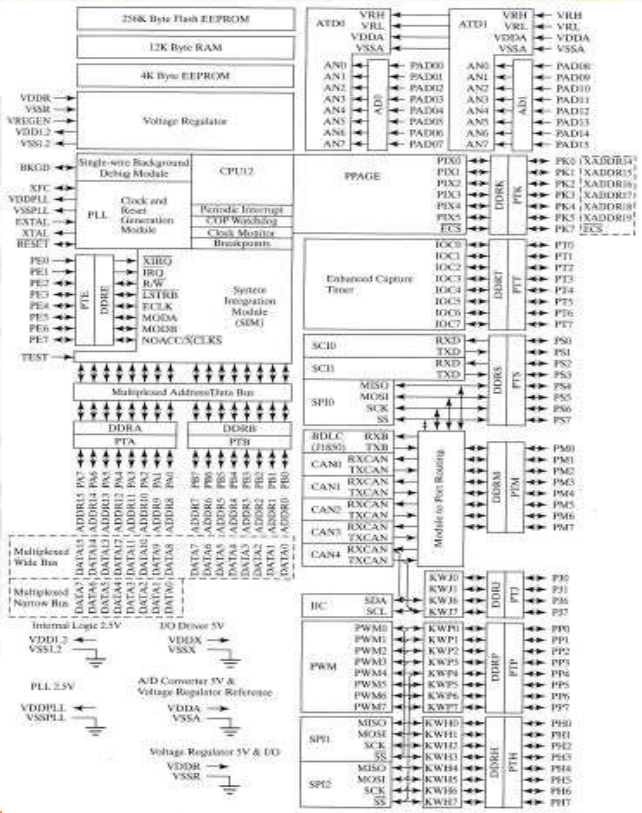
2.1 Architecture matérielle





2.1 Architecture matérielle



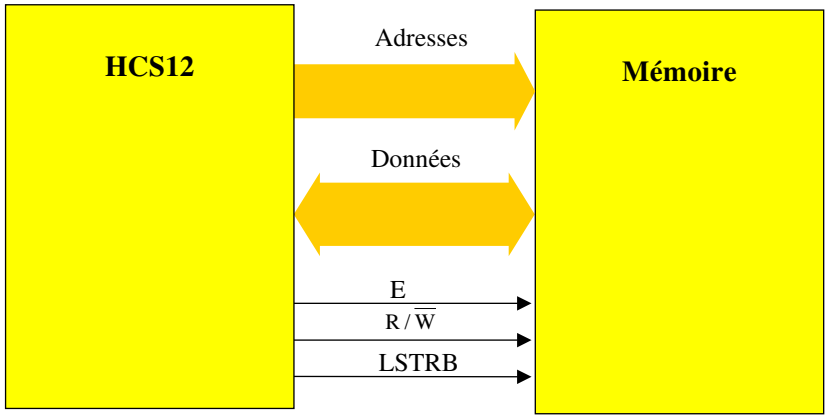


2.1 Architecture matérielle

- Bus d'adresse 16 lignes (espace adressable de 65536 octets soit 64 ko)
- Bus de données 16 lignes (accès immédiat à donnée 16 bits)
- Contrôle de la mémoire par :

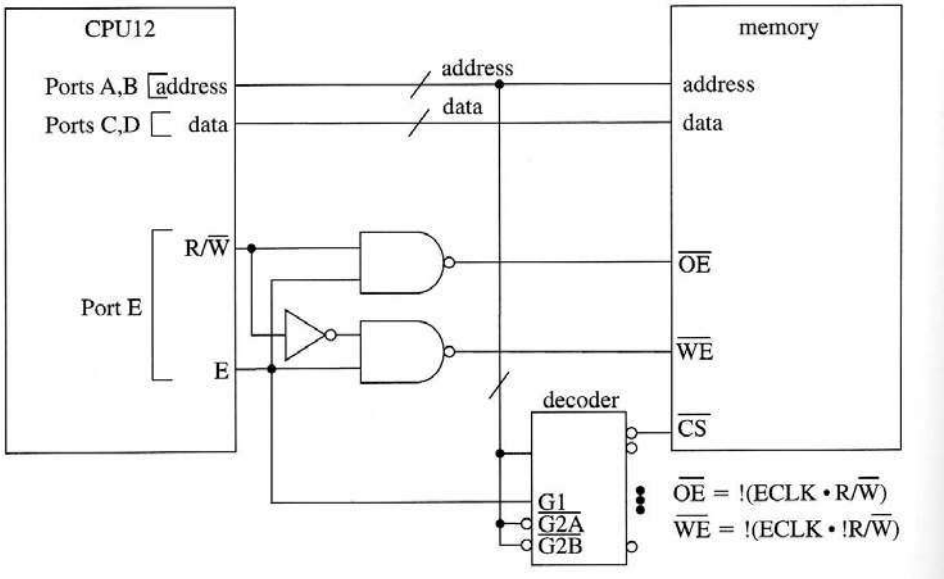
  R / $\overline{W}$ (lecture / écriture), LSTRB (accès 1 octet), E (synchronisation)

échanges entre HCS12 et mémoire sur bus non-multiplexés



2.1 Architecture matérielle

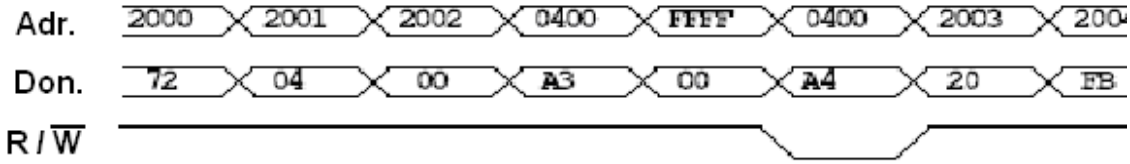
- Décodage d'adresse pour une mémoire RAM externe sur bus non-multiplexé



2.1 Architecture matérielle

• Exemple simplifié de chronogramme

Programme assembleur		Adresse : code hexadécimal
	org $2000	
loop:	inc $400	2000 : 72
		2001 : 04
		2002 : 00
	bra loop	2003 : 20
		2004 : FB



2.2 Modes d'opérations

• 8 modes possibles selon :
- états physiques de 3 broches externes : MODC (BKGD), MODB et MODA,
- ou état logiciel du registre MODE ($000B) utilisé par le bloc MEBI

MODC	MODB	MODA	Mode
0	0	0	Puce simple spéciale
0	0	1	Émulation étendue court
0	1	0	Test spécial
0	1	1	Émulation étendue large
1	0	0	Puce simple normale
1	0	1	Étendue normale court
1	1	0	Périphérique
1	1	1	Étendue normale large

À la mise sous tension mode « Puce simple normale »
(Ports A, B, K et E en entrées/sorties)

Possibilité d'écriture dans le registre MODE une seule fois

2.3 Système de mémoire

- Plan mémoire

Cas HC12 type A4

$0000	Registres internes	512 o
$0200		
$0400	Mémoire paginée Extra	1 Ko
$0800	RAM interne	1 Ko
$0C00		
$1000	EEPROM	4 Ko
$2000		
$4000		
$7000	Mémoire paginée Data	4 Ko
$8000	Mémoire paginée Program	16 Ko
$C000		16 Ko
$FF00 $FFFF	Vecteurs d'interruption	

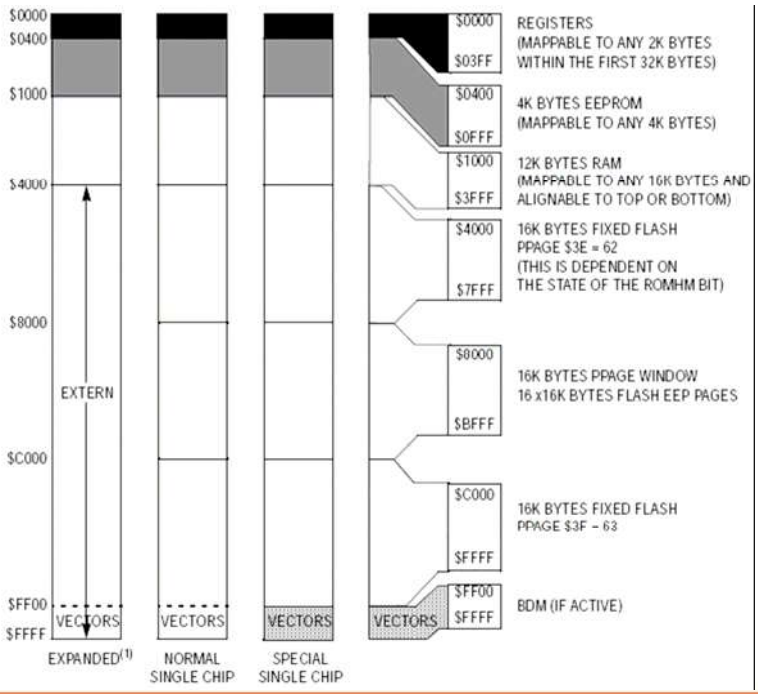
Cas HCS12 type B32

$0000	Registres internes	1 Ko
$0400	EEPROM	3 Ko
$1000	RAM interne	12 Ko
$4000	Mémoire flash EEPROM	16 Ko
$8000	Mémoire paginée Program	16 Ko
$C000	Mémoire flash EEPROM	16 Ko
$FF00 $FFFF	Vecteurs d'interruption	

2.3 Système de mémoire

- Plan mémoire selon les modes

Cas du HCS12DP256 type B32



2.3 Système de mémoire

- Plan mémoire reconfigurable par initialisation des registres
  - INITRM ($0010) : position de la RAM (12 ko)
  - INITRG ($0011) : position des registres internes (1 ko)
  - INITEE ($0012) : position de l'EEPROM (4 ko)
- Plan mémoire de base de 64 ko (bus d'adresses à 16 lignes)
- Extension mémoire par fenêtre de pagination
  - PPAGE : 1 octet pour le numéro de page

  DPAGE, EPAGE : 1 octet pour le numéro de page (*HC12 type A4*)
- Table des vecteurs d'interruption

$FFFE, $FFFF	Power-On (POR) or External Reset
$FFFC, $FFFD	Clock Monitor Reset
$FFFA, $FFFB	Computer Operating Properly (COP Watchdog Reset
$FFF8, $FFF9	Unimplemented Opcode Trap
$FFF6, $FFF7	Software Interrupt Instruction (SWI)
$FFF4, $FFF5	XIRQ
$FFF2, $FFF3	IRQ
$FFC0–$FFF1 (M68HC12)	Device-Specific Interrupt Sources
$FF00–$FFF1 (HCS12)	Device-Specific Interrupt Sources

2.4 Mémoire étendue

- Extension de la mémoire de *programme* à 1 Mo

  à travers la fenêtre $8000-$BFFF (16 ko) :
  - PPAGE ($0030) : numéro de la page 0 à 63
  - Bus d'adresses : 20 fils (A0-A13 + b0-b5 de PPAGE)
  - capacité 64 x 16 ko soit 1 Mo
- Extension de la mémoire de *données* à 1 Mo

  (*HC12 type A4*)

  à travers la fenêtre $7000-$7FFF (4 ko)
  - DPAGE : numéro de la page 0 à 255
- Extension de la mémoire *extra* à 256 ko

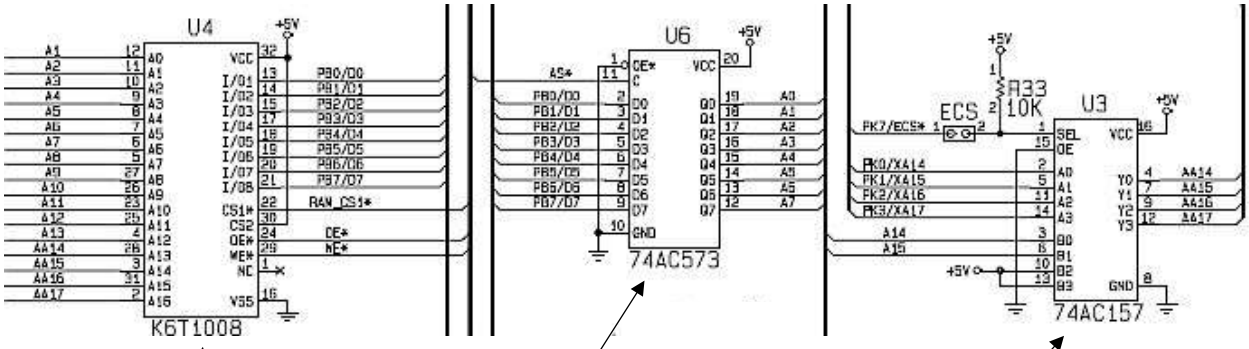
  (*HC12 type A4*)

  à travers la fenêtre $0400-$07FF (1 ko)
  - EPAGE : numéro de la page 0 à 255

2.4 Mémoire étendue

- Décodage d'adresse

Portion de la carte HCS12 utilisée en TP

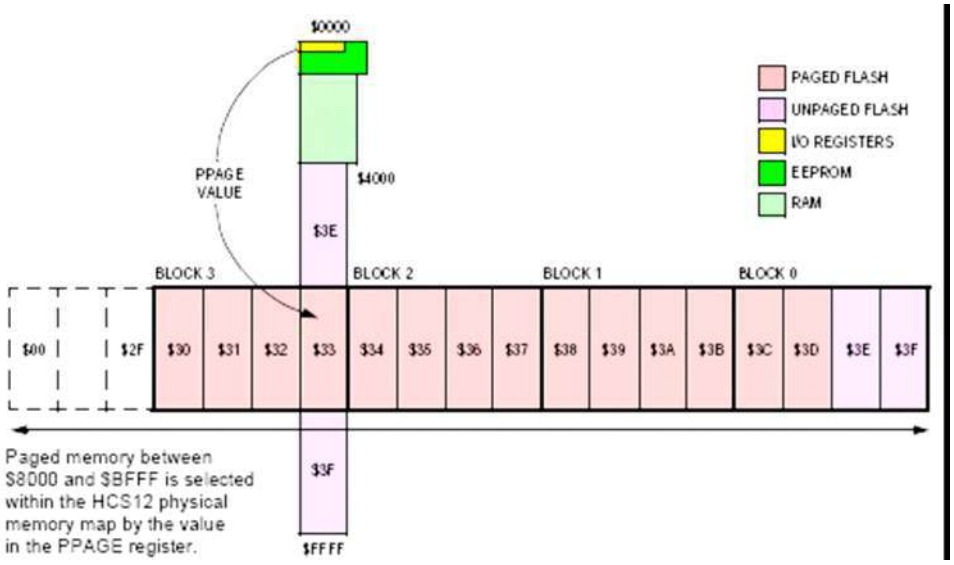


Mémoire RAM *Mémoire tampon adresse* *Sélection entrée A ou B*

PPAGE0 à PPAGE5 = PK0 à PK5 = XA14 à XA19

2.4 Mémoire étendue

- Exemples d'extension de la mémoire de *programme* avec PPAGE = $33



2.4 Mémoire étendue

• Exemples d'extension de la mémoire de *programme*

Bank	Window	A15 . . . A0	Bank + A13 . . . A0
0	8000–BFFF	0000–3FFF	00000–03FFF
1	8000–BFFF	4000–7FFF	04000–07FFF
2	8000–BFFF	8000–BFFF	08000–0BFFF
3	8000–BFFF	C000–FFFF	0C000–0FFFF
4	8000–BFFF	0000–3FFF	10000–13FFF
.	.	.	.
.	.	.	.
.	.	.	.
3B	8000–BFFF	C000–FFFF	EC000–EFFFF
3C	8000–BFFF	0000–3FFF	F0000–F3FFF
3D	8000–BFFF	4000–7FFF	F4000–F7FFF
3E	8000–BFFF	8000–BFFF	F8000–FBFFF
3F	8000–BFFF	C000–FFFF	FC000–FFFFF

• Bits b0-b5 de PPAGE = banc ou page 0 à 63 (XADR14-XADR19)
ex. : adresse linéaire $EC000 -> banc $3B et adresse $8000

2.4 Mémoire étendue

• Configuration logicielle du HCS12 en mode étendu

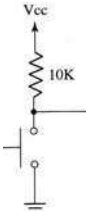
- Mode normal à la mise sous tension
- Initialisation du registre MODE à $E3 (mode étendu large)
- Initialisation du registre PEAR (Port E Assignment) à $0C :
  b4 NECLK à 0 pour valider la broche 4 en sortie d'horloge ECLK
  b3 LSTRE à 1 pour valider la broche 3 en sortie LSTRB
  b2 RDWE à 1 pour valider la broche 2 en sortie RW
- Initialisation du registre MISC (Miscellaneous) à $01 :
  pas d'extension de durée d'horloge
  b1 ROMHM à 0 pour activer la flash EEPROM en $4000
  b0 ROMON à 1 pour valider la flash EEPROM
- Initialisation du registre EBICTL (External Bus Interface Control) à $00 :
  pas d'élongation de la période d'horloge

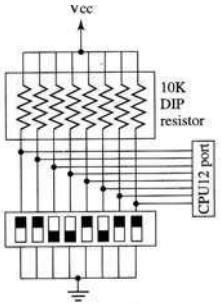
2.4 Mémoire étendue

- Configuration matérielle du HCS12 (type B32) en mode étendu
  - ➢ Gestion de l'accès par le sous-bloc MEBI (Multiplexed External Bus Interface) avec MODE, EBICTL, et PEAR
  - ➢ Gestion de la mémoire par le sous-bloc MMC (Module Mapping Control) avec MISC, PPAGE, INITRM, INITEE, INITRG
  - ➢ PPAGE sur PORTK (b0 à b5), adresses/données sur PORTA et PORTB
  - ➢ Synchronisation par ECLK, RW et LSTRB

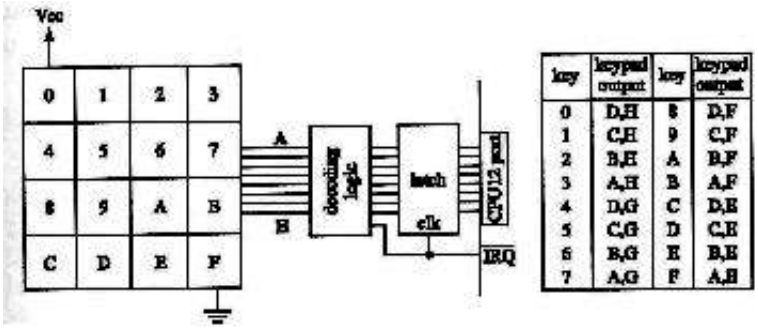
2.5 Applications

- Entrées :
  - ➢ Interrupteurs





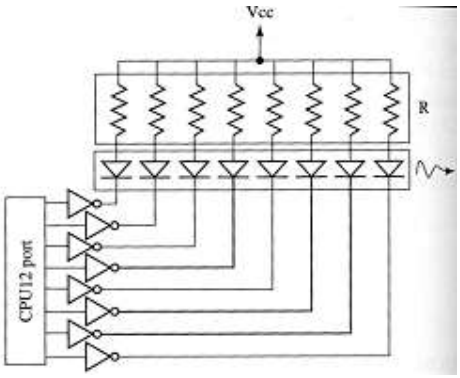
  - ➢ Claviers



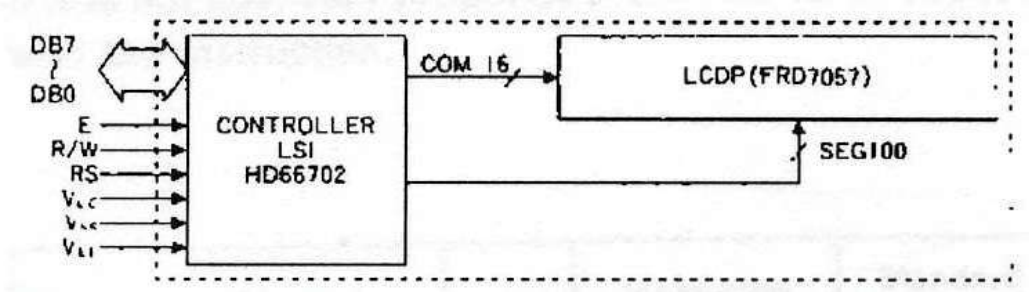
key	keypad output	key	keypad output
0	D,H	8	D,F
1	C,H	9	C,F
2	B,H	A	B,F
3	A,H	B	A,F
4	D,G	C	D,E
5	C,G	D	C,E
6	B,G	E	B,E
7	A,G	F	A,E

2.5 Applications

- Sorties :
  - LED



  - LCD



2.6 Tests

1. Donnez l'architecture matérielle du HCS12.

2. Que doit faire l'utilisateur pour placer le microcontrôleur en mode puce simple normale ?

3. Quelle est la taille de la fenêtre de pagination pour étendre la mémoire programme ?

4. Donnez les principales caractéristiques du HCS12.

5. Combien de ports a le MC9S12DP256 ?

6. Quel est le mode principal du MC9S12DP256 ?

Chapitre 3 : Mémoires - PAL

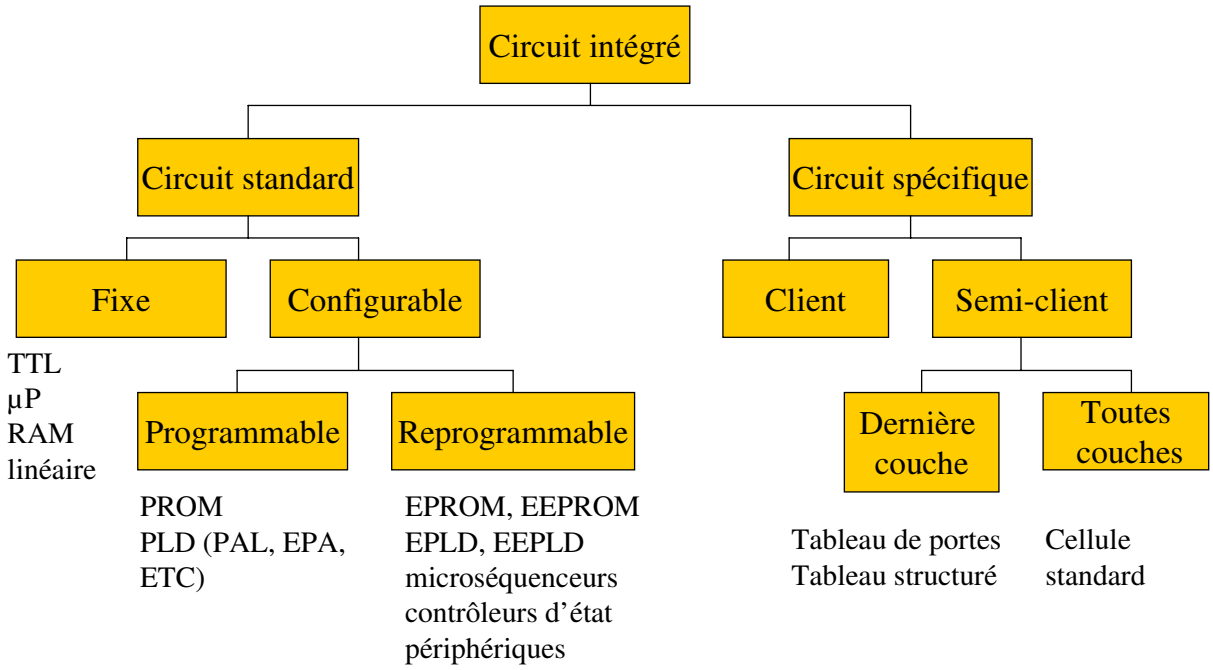
- Sommaire

3.1 Introduction

- Types principaux :
  - RAM (random access memory)
    - mémoire volatile
    - écriture ou lecture
    - temps d'accès plus rapide
    - utilisation pour stockage temporaire de donnée
  - ROM (read only memory)
    - mémoire non volatile
    - lecture seule
    - temps d'accès plus lent
    - utilisation pour stockage d'instructions et de constantes pendant l'exécution de programmes

3.1 Introduction

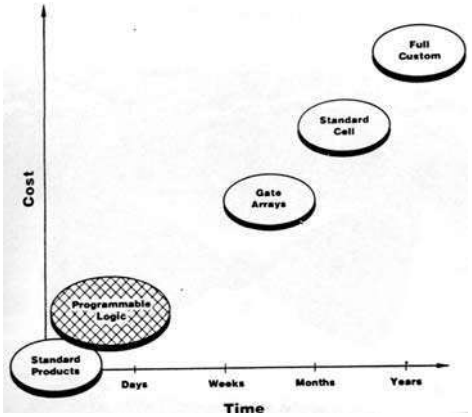
- Famille de circuit intégré



3.1 Introduction

TABLE 1-1. IC SELECTION CRITERIA FOR DIFFERENT ARCHITECTURES

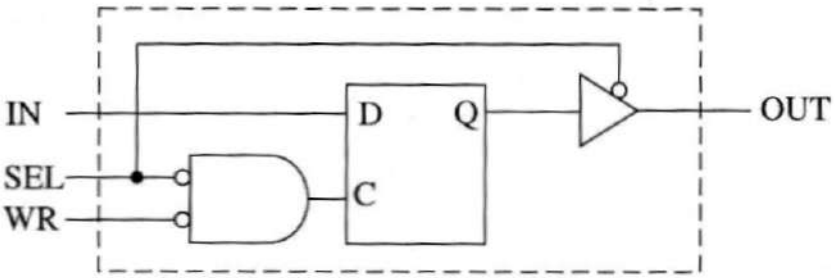
Criteria	Standard Products		Gate Arrays/ Standard Cells	Programmable Logic
	TTL SSI/MSI	LSI		
Development Lead Time	Immediate	Immediate	Weeks/Months	Hours
Development Cost	None	None	≈ $20K	Low
Second Sources	Many	Several	Few	Many
Architectural Flexibility	Medium	Low	High	Medium
Logic Density	Low	High	Medium/High	Medium
IC Design Expertise Required	None	None	Some/Much	None



3.2 RAM

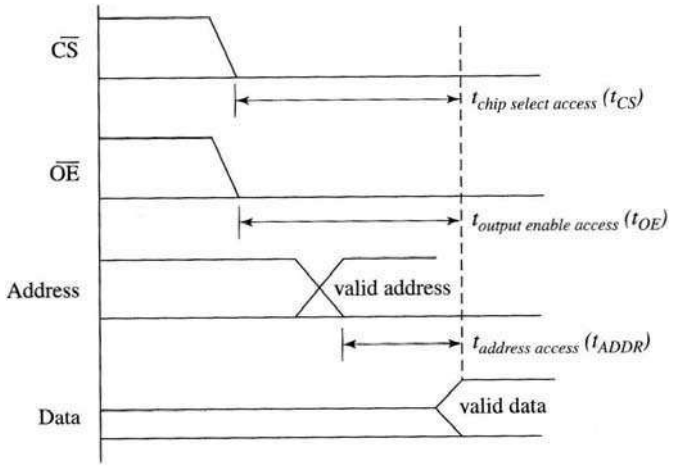
- Static RAM : cellule de base à plusieurs transistors,
  accès rapide, consomme plus, coûteux

- Dynamic RAM : cellule de base à un transistor,
  densité forte, accès lent,
  rafraîchissement nécessaire
- Lignes de contrôle : WE (write enable), OE (output enable),
  CS (chip select)
- Cellule de base : bascule D



3.2 RAM

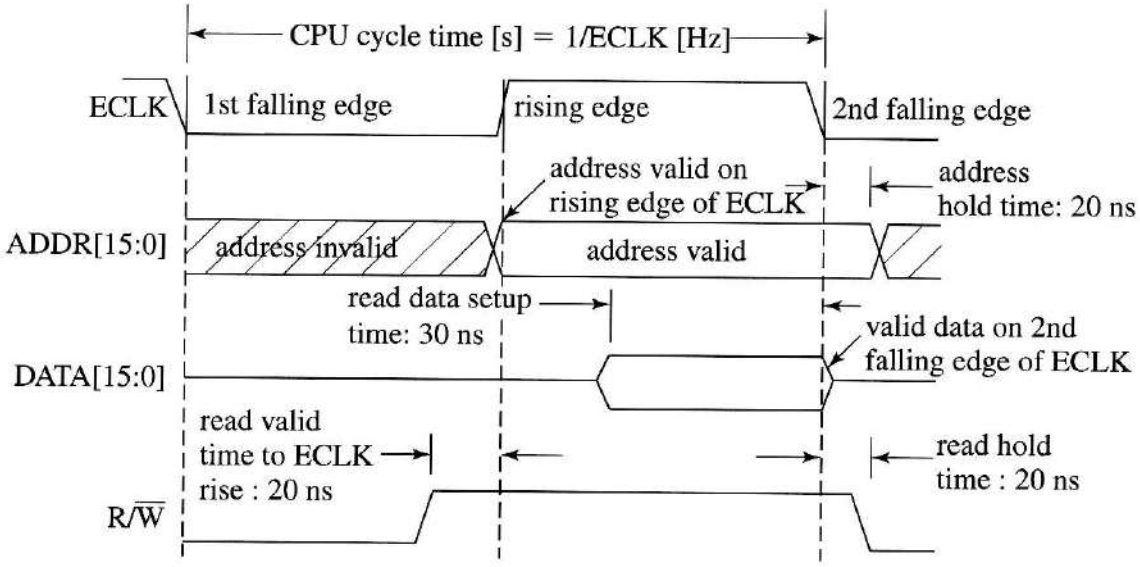
- Temps d'accès : adresse (t_{ADDR}), sélection (t_{CS}), sortie (t_{OE})



- Interface avec le HCS12 :
  étude de compatibilité des temps d'accès

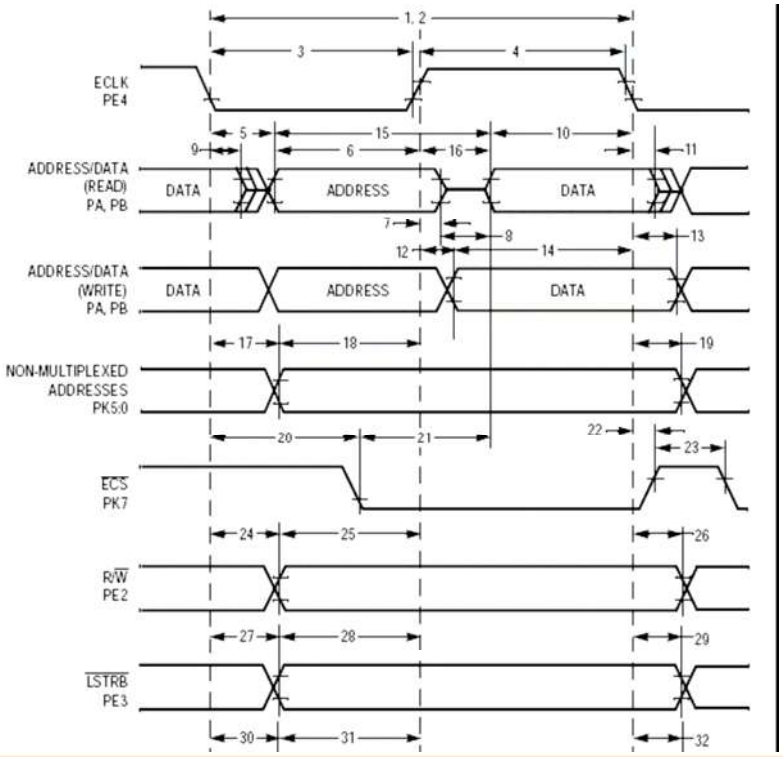
3.2 RAM

- Cycle de lecture du HC12 (bus non multiplexé)



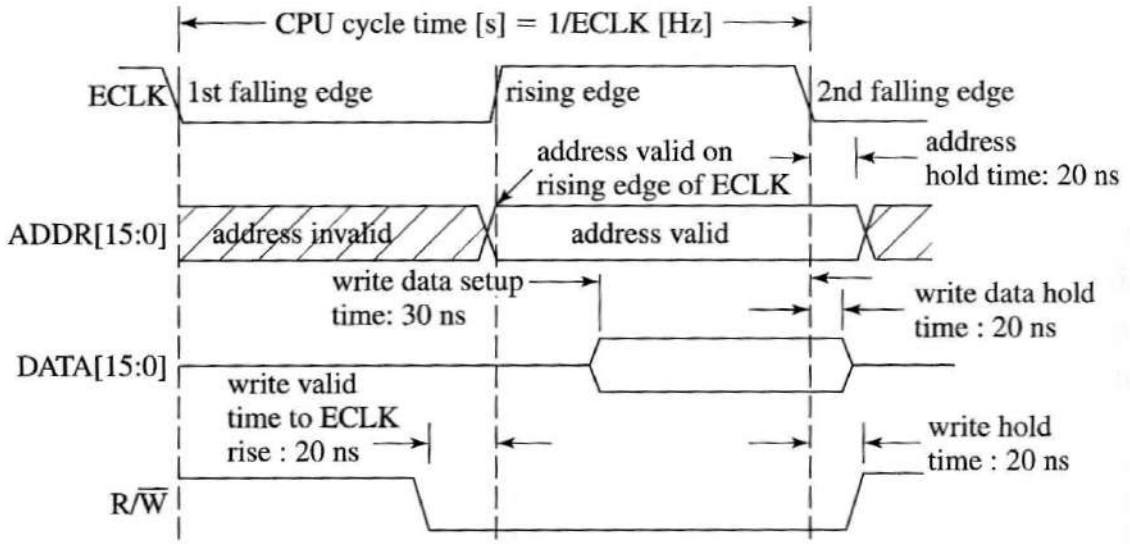
3.2 RAM

- Cycle de lecture du HCS12 (bus multiplexé)



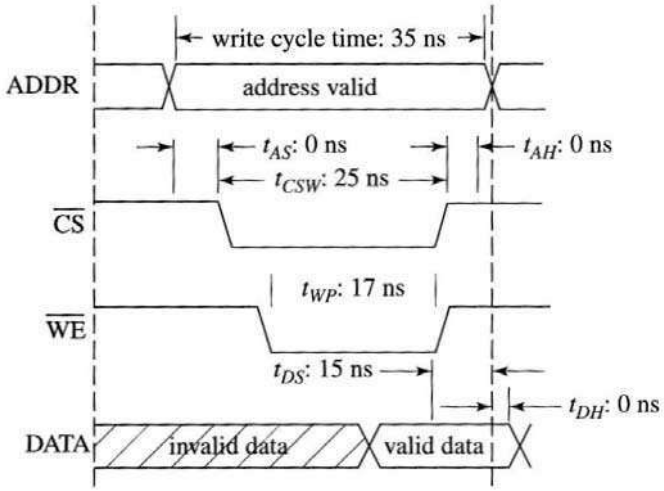
3.2 RAM

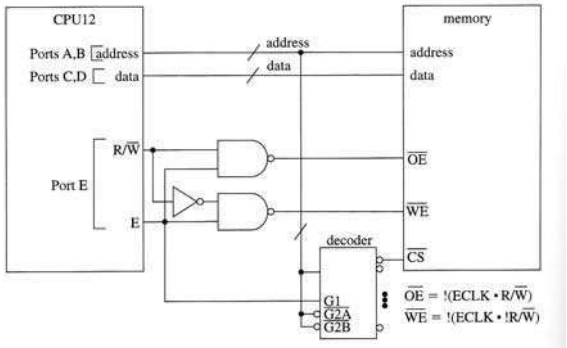
- Cycle d'écriture du HC12 (bus non multiplexé)



3.2 RAM

- Cycle d'écriture d'une RAM 35 ns





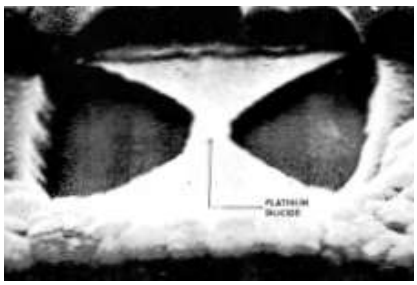
3.3 ROM

- ROM : lecture seule, mémoire non modifiable
- PROM : programmable 1 fois (tableau de fusibles)
- EPROM : programmable électriquement, effacement par UV
- EEPROM : programmation électrique,
  effacement électrique pour chaque octet (type *simple*)
  ou pour un bloc d'octets (type *flash*)
- Tableau de fusibles par combinaison de matrice de portes OU et de matrice de portes ET
  FPLA (Field Programmable Logic Array)
  PAL (Programmable Array Logic)

	ET	OU	options de sortie
PROM	Fixe	Prog.	S
PAL	Prog.	Fixe	E / S mémoire
FPLA	Prog.	Prog.	E / S mémoire

3.3 ROM

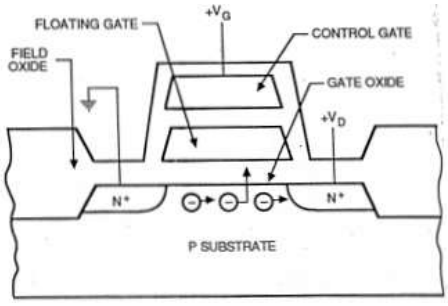
- Fusible



Avant programmation

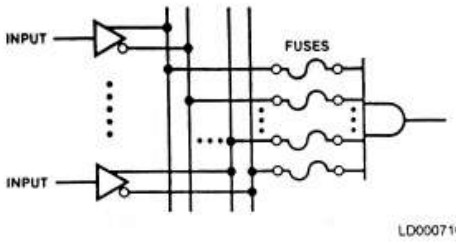
Après programmation

- Contrôle

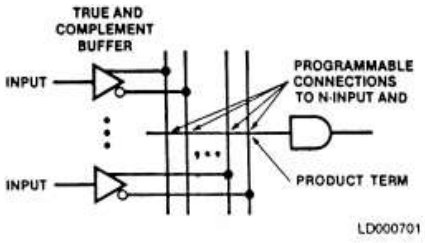


3.3 ROM

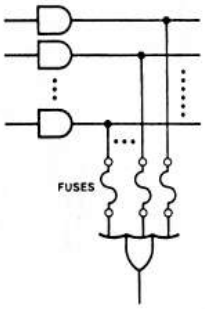
- Matrices de OU et ET



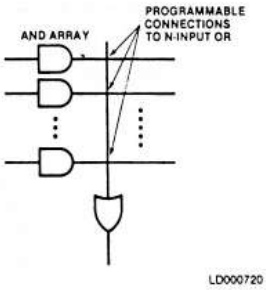
Programmable-AND Array Logic Equivalent



Programmable-AND Array Logic Diagram Notation



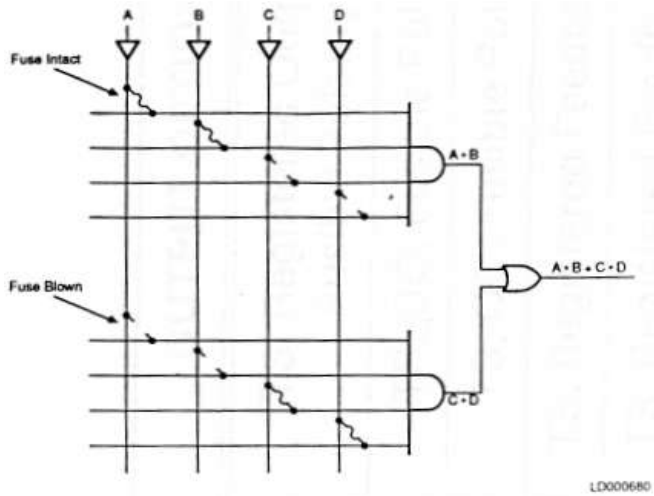
Programmable-OR Array Logic Equivalent



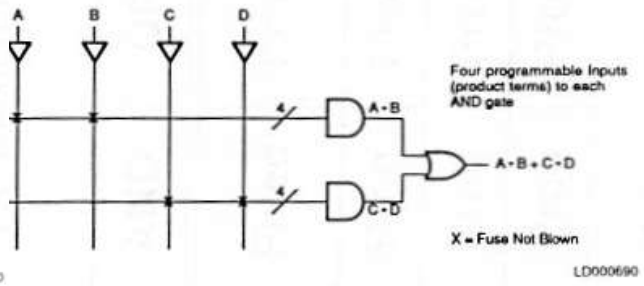
Programmable-OR Array Logic Diagram Notation

3.3 ROM

- Matrices de OU et ET



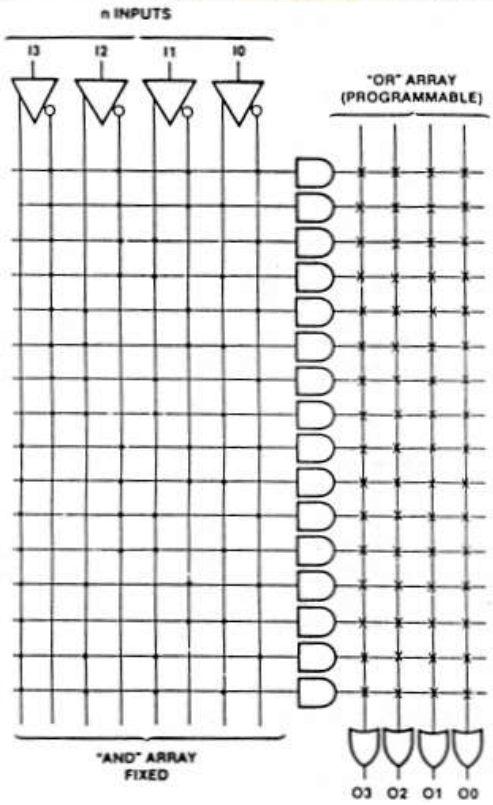
Fuse Implementation of AND-OR



Conventional Representation

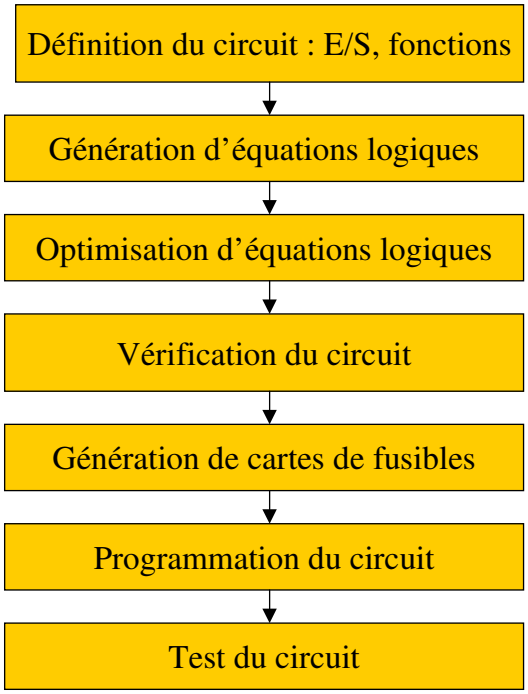
3.3 ROM

- PROM



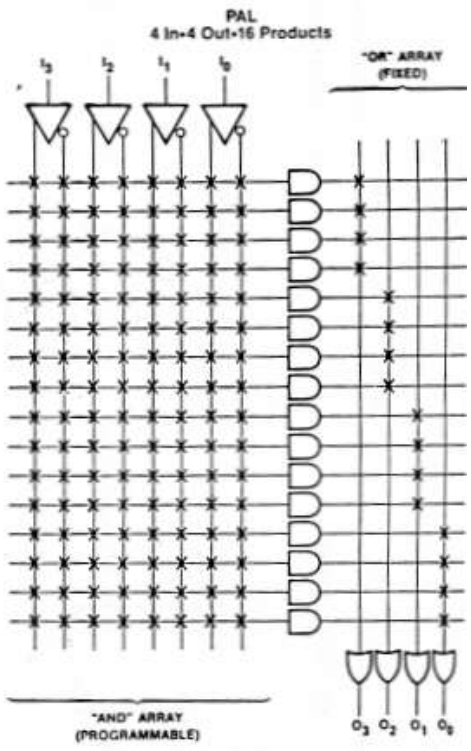
3.4 PAL

- Procédure de programmation de PAL



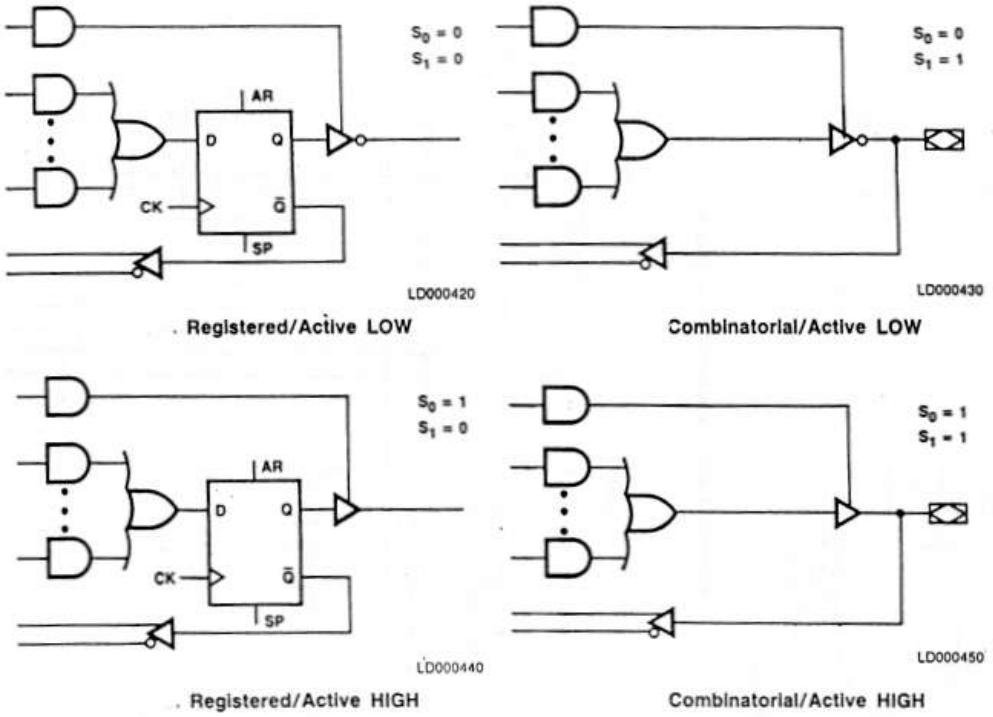
3.4 PAL

- Matrice de fusibles



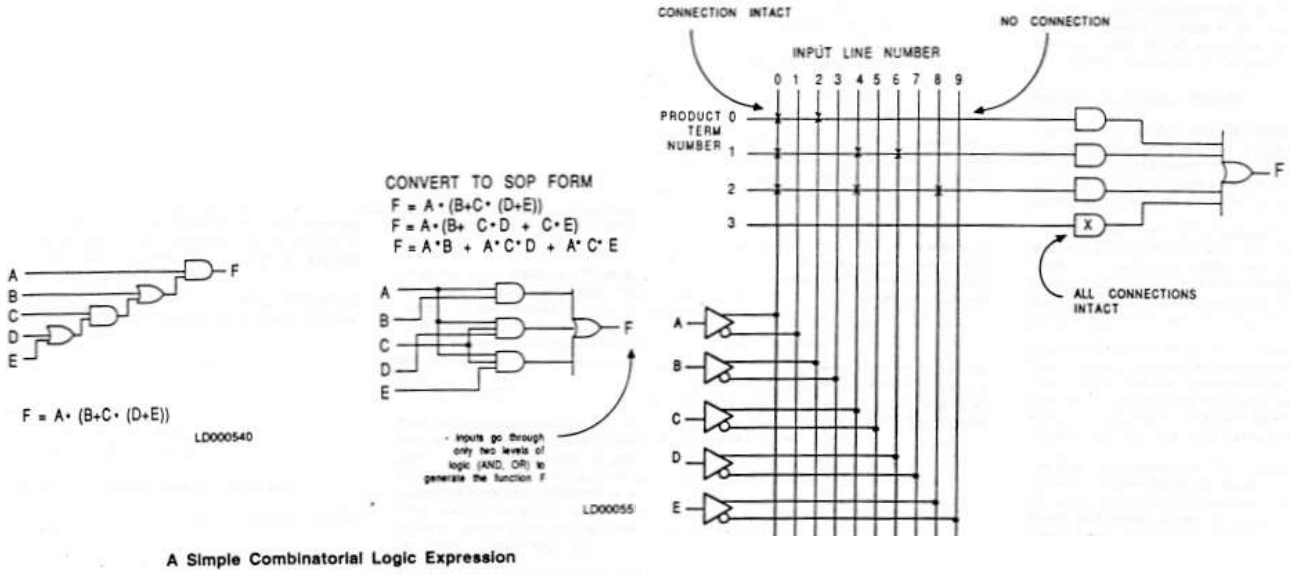
3.4 PAL

- Options de sortie



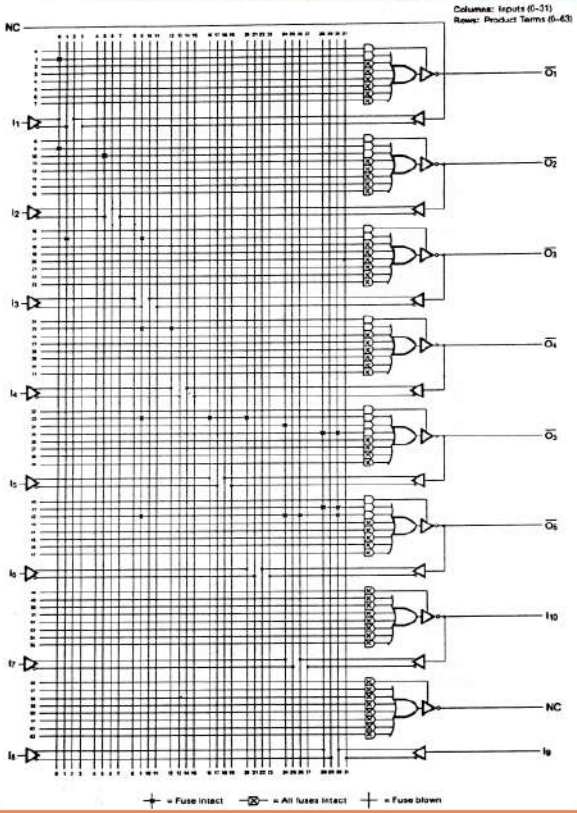
3.5 Applications

- Programmation de PAL



3.5 Applications

- AmPAL16L8

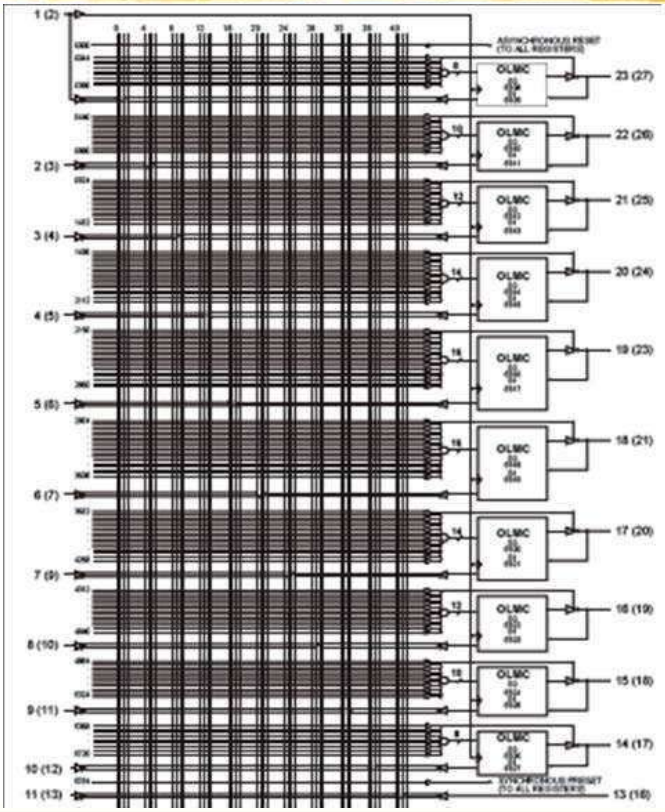


3.5 Applications

- AmPAL22V10

  10 sorties

  macrocellule logique en sortie



3.5 Applications

- Exemple de composants décrits en VHDL
  (Very high-speed integrated circuit Hardware Description Language)
  - Porte ET à 2 entrées

```
ENTITY TwoStateNand2 IS
      PORT (a, b, outen: IN BIT;  c: OUT BIT);
END TwoStateNand2;
ARCHITECTURE show OF TwoStateNand2 IS
      SIGNAL temp: BIT;
      BEGIN
      CALCUL : PROCESS
      BEGIN
            temp <= a NAND b;
            IF outen='1' THEN c <= temp;
            ELSE  c <= '1';
            END IF;
      END PROCESS CALCUL;
END show;
```

  - Bascule D commandée sur front montant

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY dff  IS
      port (data, clk : in std_logic; q : out std_logic);
END dff;
ARCHITECTURE behav OF dff IS
       BEGIN
            process (clk)
             BEGIN
                              IF (clk'event and clk = '1') THEN q <= data;
                              END IF;
             END PROCESS;
END behav;
```

3.6 Tests

1. Expliquez les différences entre les ROMs, RAMs, EEPROMs et PALs.

2. Un système de mémoire exige 4 millions d'unités. Dans chaque unité est stocké un mot de 16 bits. Spécifiez le nombre de lignes d'adresses, le nombre de ligne de données et la taille de la mémoire en octets.

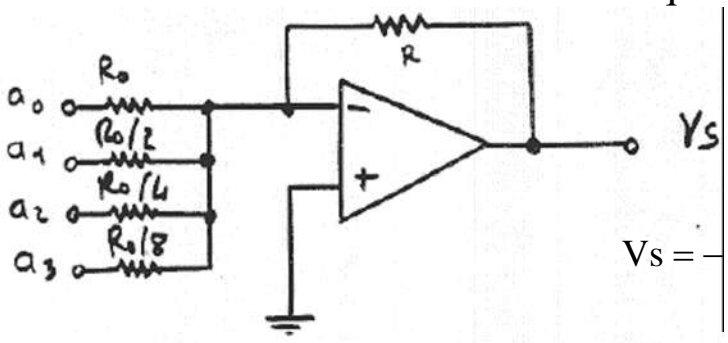
3. On souhaite réaliser une analyse complète de mémoire pour une EPROM de 32 Ko située à l'adresse $8000. En supposant que la fréquence de l'horloge du HCS12 est de 4 MHz, spécifiez les temps d'accès de l'EPROM.

Chapitre 4 : Convertisseurs CNA et CAN

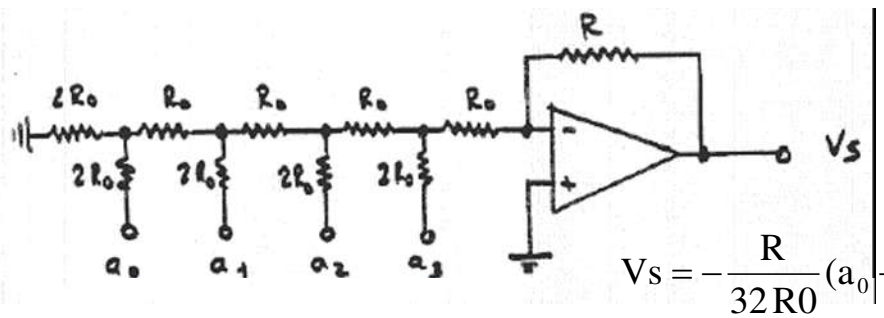
- Sommaire

4.1 CNA

- Schéma de convertisseurs numériques analogiques



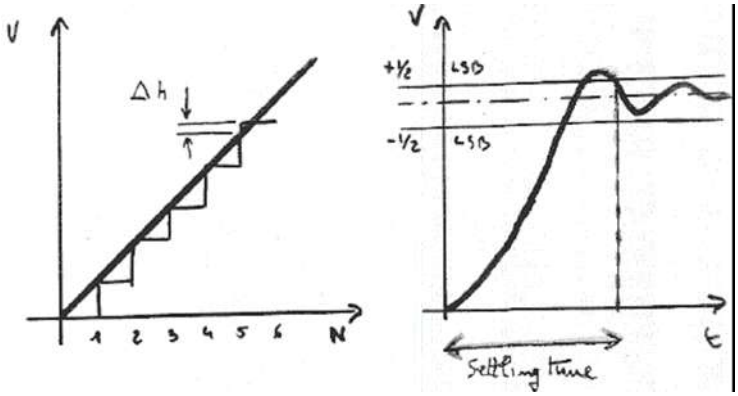
$$Vs = -\frac{R}{R0}(a_0 + 2a_1 + 4a_2 + 8a_3)E$$



$$Vs = -\frac{R}{32\,R0}(a_0 + 2a_1 + 4a_2 + 8a_3)E$$

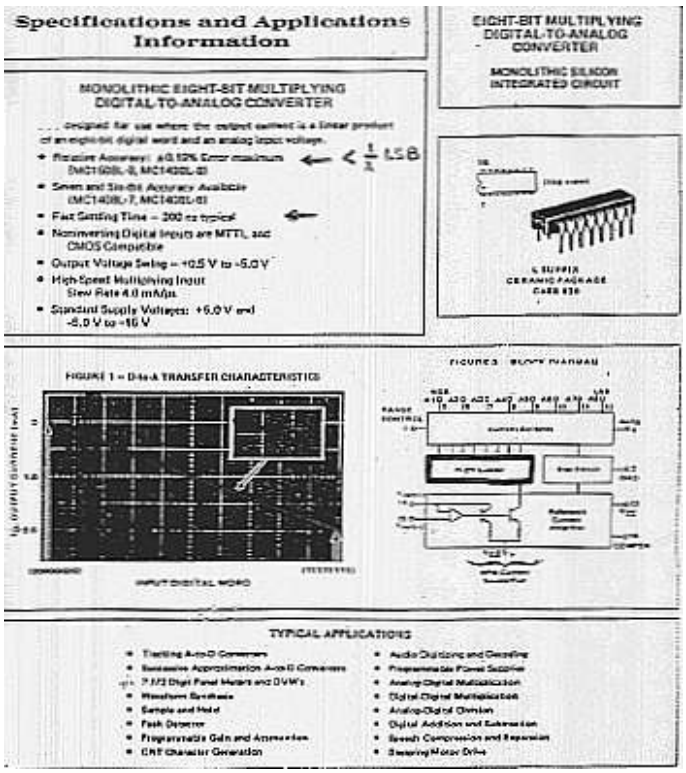
4.1 CNA

- Caractéristiques
  - Résolution (8 à 16 bits)
  - Linéarité mieux que ½ LSB
  - Précision
  - Temps d'établissement (settling time)



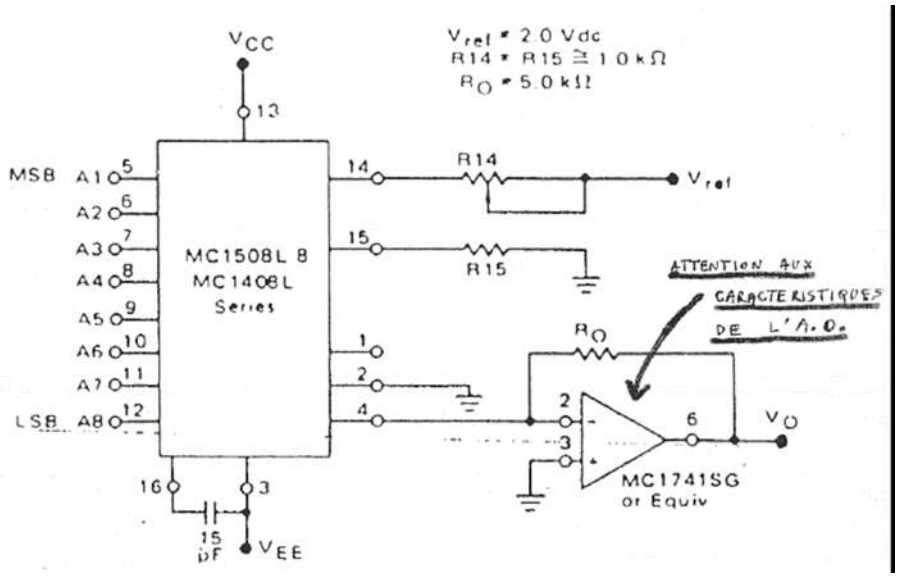
4.1 CNA

- CNA MC1408L8



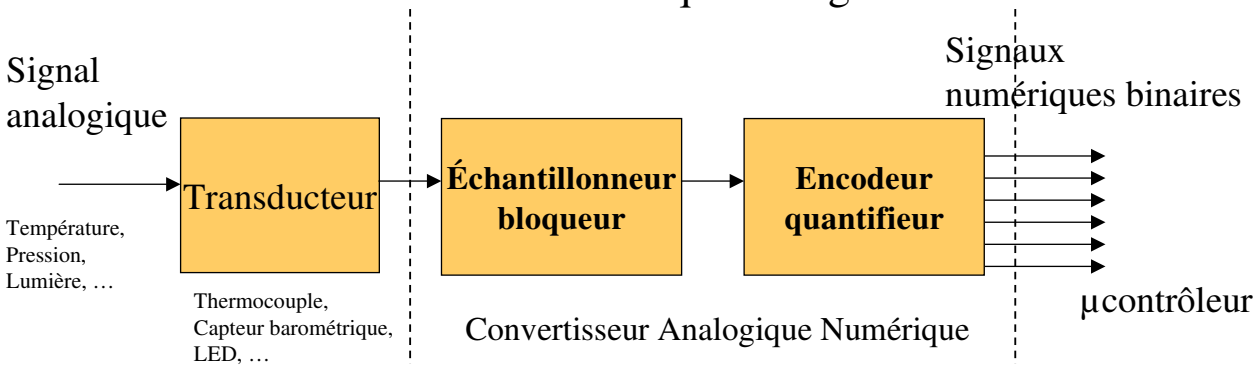
4.1 CNA

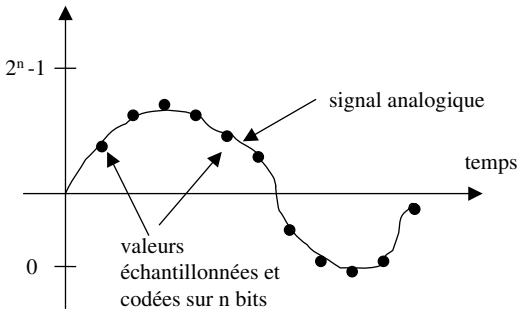
- Montage standard



4.2 Introduction au CAN

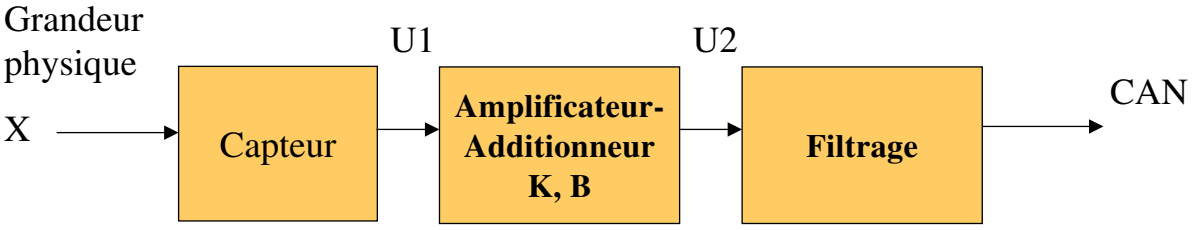
- 1er maillon du traitement numérique du signal





4.3 Transducteur

- Caractéristiques



U2 = K U1 + B

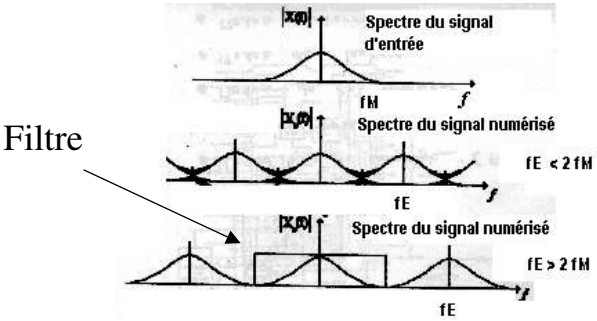
K : gain

B : offset

• Calibration pour avoir la pleine échelle en entrée du CAN

4.3 Transducteur

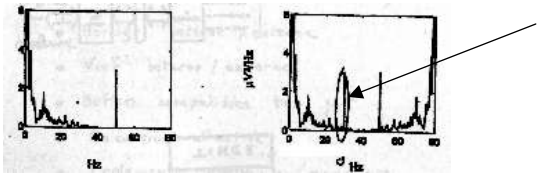
- Filtrage anti-repliement



fM : fréquence maximale du signal d'entrée

fE : fréquence d'échantillonnage du CAN

- Exemples de spectre du signal numérisé



Spectre original Spectre après numérisation

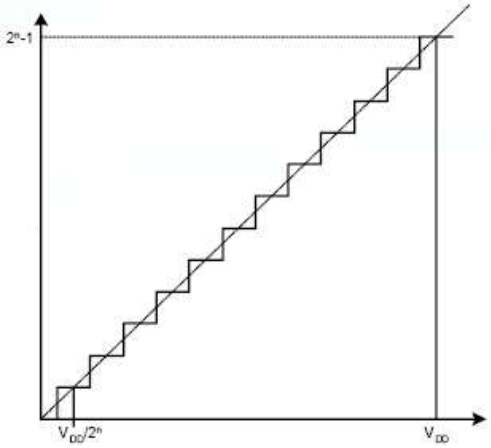
4.4 Caractéristiques du CAN

- Caractéristique idéale du convertisseur

  - un CAN n-bits a une sortie binaire à 2^n valeurs

  - l'écart entre les deux droites ci-dessous est l'erreur de quantification

sortie codée



tension analogique

  - un CAN peut avoir des erreurs de linéarité et de monotonicité

4.4 Caractéristiques du CAN

- Caractéristiques
  - tensions de référence en entrée : VRL (basse), VRH (haute)
  - nombre n de bits
  - résolution R en tension

    $R = (VRH - VRL) / 2^n$
  - gamme dynamique en dB

    $G = 20 \log 2^n$
  - fréquence d'échantillonnage fE

    $fE > 2\ fM$ (théorème de Shannon)
  - temps de conversion
  - débit binaire

    $D = n\ fE$
  - isolement analogique / numérique
  - réglage de gain et de décalage

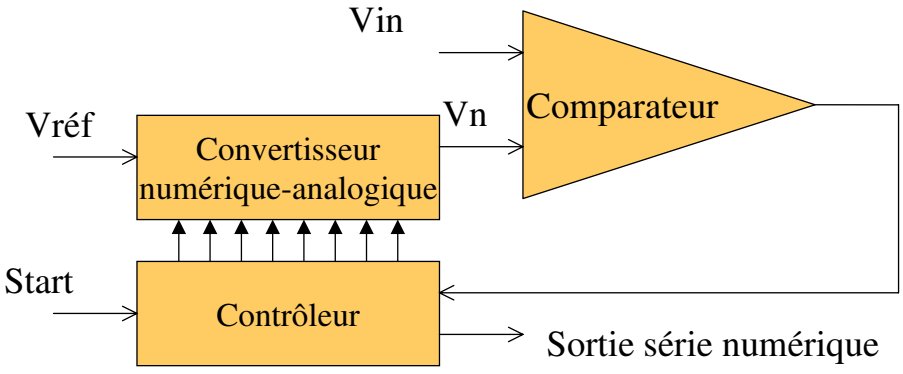
4.5 Technologies du CAN

- Types principaux
  - Approximations successives
  - Intégration
  - Comptage
  - Parallèle

Le HCS12 utilise un CAN 10 bits de type à approximations successives.

4.5 Technologies du CAN

- Convertisseurs à approximations successives

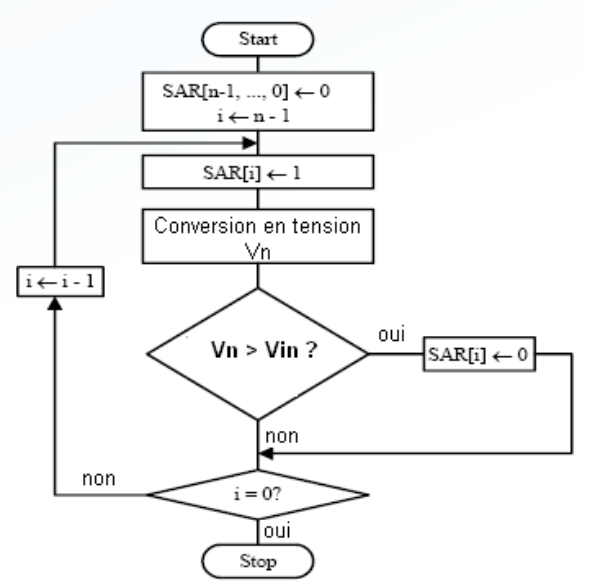


Le convertisseur approxime une tension d'entrée Vin en une tension Vn en n essais. Le contrôleur utilise un registre à décalage SAR (successive approximation register) en entrant 0 ou 1 selon le résultat de la comparaison.

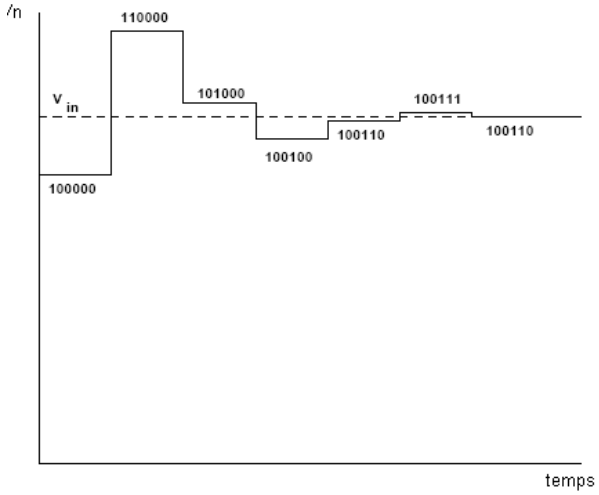
4.5 Technologies du CAN

- Convertisseurs à approximations successives

Algorigramme

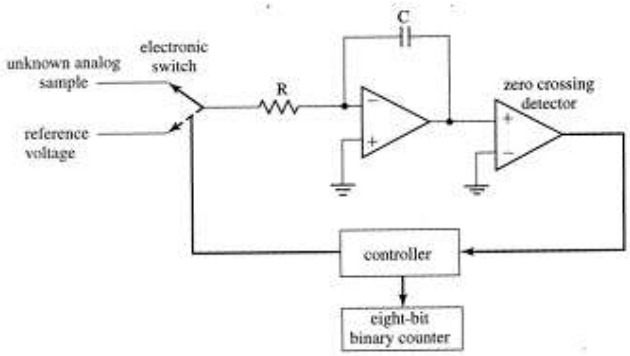


Chronogramme



4.5 Technologies des convertisseurs

- Convertisseurs à intégrateurs
  - utilisation d'une capacité pour charge et décharge
  - mise en œuvre simple
  - convertisseur adapté aux applications lentes
  - cas du microcontrôleur Microchip PIC 14400



Principe du convertisseur double pente

4.5 Technologies des convertisseurs

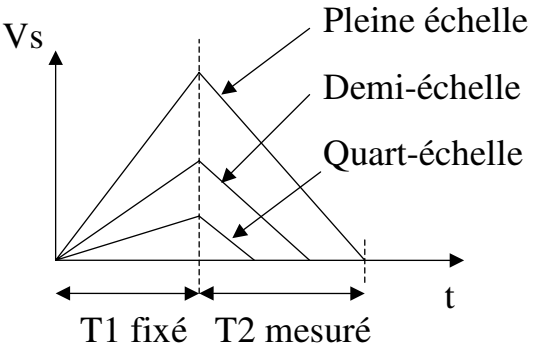
- Convertisseurs à intégrateurs

  - équations pour une tension de référence Vref et une tension inconnue Vx

$$0 < t < T1 : \quad Vs = \frac{Vx\ t}{RC}$$

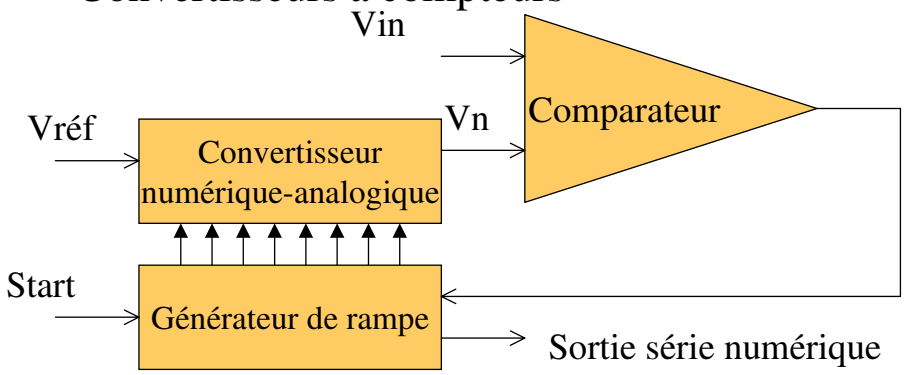
$$T1 < t < T2 : \quad Vs = -\frac{Vref\ (t - T1)}{RC} + \frac{Vx\ T1}{RC}$$

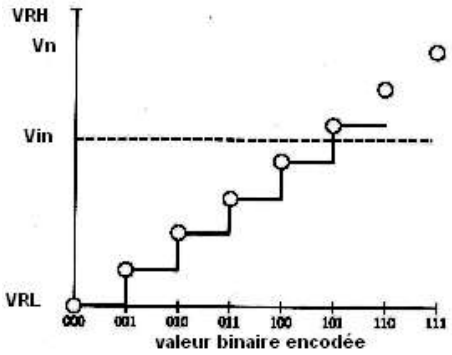
$$Vx = \frac{Vref\ T2}{T1}$$



4.5 Technologies des convertisseurs

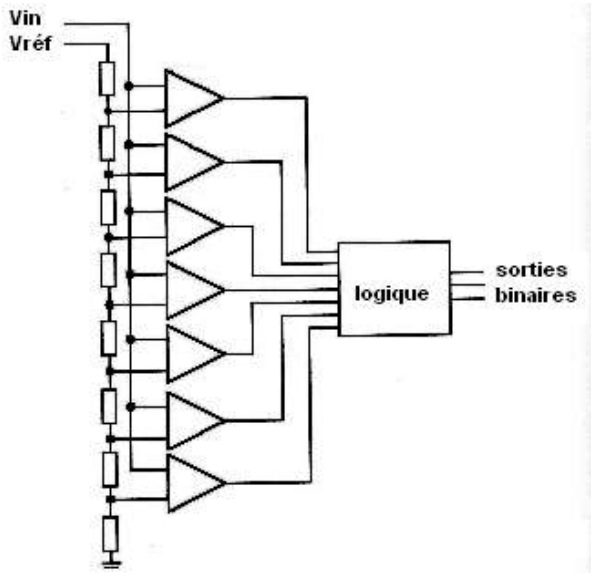
- Convertisseurs à compteurs





4.5 Technologies des convertisseurs

- Convertisseurs parallèles
  - 2^n-1 comparateurs
  - Conversion rapide
  - Coût élevé

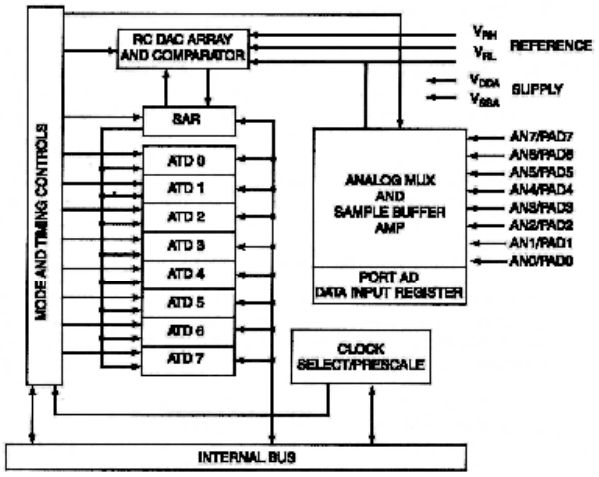


4.6 Cas du HCS12

- Caractéristiques du MC9S12DP256B
  - 2 CANs 10 bits 8 canaux (AD0 et AD1)
  - Horloge de fréquence 500 kHz à 2 MHz
  - Résolution 8 ou 10 bits
  - Temps typique de conversion de 7 µs
    pour n = 10 bits et fE = 2 MHz
  - Résultat numérique signé ou non signé,
    justifié à droite ou à gauche
  - Erreur de quantification de ± 1 LSB
  - Acquisition canal seul ou multi-canaux en séquentiel
  - Déclenchement externe possible

4.6 Cas du HCS12

- Architecture de AD0



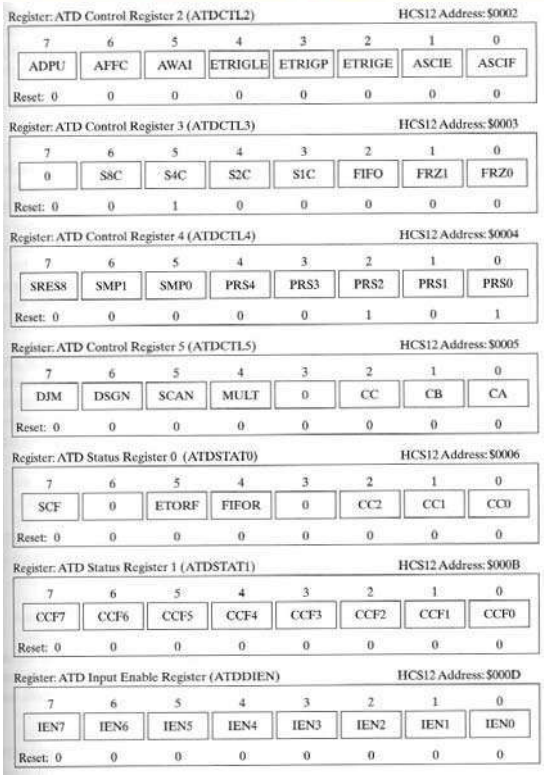
- Brochage
  - Broches AN0 à AN7 (module AD0)
  - Broches AN8 à AN15 (module AD1)
  - Broche AN7 ou AN15 configurable en signal de déclenchement
  - V_{RH} et V_{RL} : tensions de référence haute et basse
  - V_{DDA} et V_{SSA} : tension d'alimentation et masse pour le convertisseur.

4.6 Cas du HCS12

- Chaque module CAN comprend les registres :
  - 6 registres de contrôle : ATDxCTL0 à ATDxCTL5 (x = 0 ou 1)
    ATDxCTL0 et ATDxCTL1 sont seulement utilisés en usine
  - 2 registres d'état : ATDxSTAT0 et ATDxSTAT1
  - 2 registres de test : ATDxTEST0 et ATDxTEST1
  - 1 registre de validation des entrées : ATDxDIEN
  - 1 registre 8 bits de port de données : PORTADx
  - 8 registres 16 bits de résultat : ATDxDR0 à ATDxDR7 (Low et High)

4.6 Cas du HCS12

- Registres d'un CAN :



Register: ATD Control Register 2 (ATDCTL2) — HCS12 Address: $0002

7	6	5	4	3	2	1	0
ADPU	AFFC	AWAI	ETRIGLE	ETRIGP	ETRIGE	ASCIE	ASCIF
Reset: 0	0	0	0	0	0	0	0

Register: ATD Control Register 3 (ATDCTL3) — HCS12 Address: $0003

7	6	5	4	3	2	1	0
0	S8C	S4C	S2C	S1C	FIFO	FRZ1	FRZ0
Reset: 0	0	1	0	0	0	0	0

Register: ATD Control Register 4 (ATDCTL4) — HCS12 Address: $0004

7	6	5	4	3	2	1	0
SRES8	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
Reset: 0	0	0	0	0	1	0	1

Register: ATD Control Register 5 (ATDCTL5) — HCS12 Address: $0005

7	6	5	4	3	2	1	0
DJM	DSGN	SCAN	MULT	0	CC	CB	CA
Reset: 0	0	0	0	0	0	0	0

Register: ATD Status Register 0 (ATDSTAT0) — HCS12 Address: $0006

7	6	5	4	3	2	1	0
SCF	0	ETORF	FIFOR	0	CC2	CC1	CC0
Reset: 0	0	0	0	0	0	0	0

Register: ATD Status Register 1 (ATDSTAT1) — HCS12 Address: $000B

7	6	5	4	3	2	1	0
CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
Reset: 0	0	0	0	0	0	0	0

Register: ATD Input Enable Register (ATDDIEN) — HCS12 Address: $000D

7	6	5	4	3	2	1	0
IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0
Reset: 0	0	0	0	0	0	0	0

4.6 Cas du HCS12

- Fonctions des registres de contrôle :

  - ATDxCTL2 ($62) :
    - mise en service du convertisseur (ADPU)
    - mise à zéro des drapeaux (AFFC)
    - mode d'attente (AWAI)
    - validation et polarité du déclenchement (ETRIGE, ETRIGP)
    - validation et drapeau d'interruption (ASCIE, ASCIF)

  - ATDxCTL3 ($63) :
    - suspension de la conversion (FRZ0, FRZ1)
    - utilisation cyclique des 8 registres de résultats (FIFO)
    - nombre limite de conversions de 1 à 8 (S8C, S4C, S2C, S1C)

4.6 Cas du HCS12

- Fonctions des registres de contrôle :
  - ATDxCTL4 ($64) :
    - diviseur de fréquence de l'horloge de base BusClock
      nombre N 5 bits (PRS0 à PRS4) à ajuster tel que :
      500 kHz < fCAN = 0,5 f BusClock / (N+1) < 2 MHz
    - programmation du temps d'échantillonnage de 2 à 16 périodes d'horloge
      2 bits (SMP0, SMP1)
    - résolution Nr = 8 ou 10 bits (SRES8)
      temps de conversion = tCAN (2 + Nr + Nsmp)

  - ATDxCTL5 ($65) :
    - choix du mode de conversion (simple, continu, multi-canaux) (SCAN, MULT)
    - sélection du canal à convertir (CC, CB, CA)
    - résultat signé ou non (DSGN), justifié à droite ou à gauche (DJM)

4.6 Cas du HCS12

- Fonctions des registres d'état
  - Chaque bit de ATDxSTAT1 correspond à un drapeau d'état : une valeur à 1 correspond à une fin de conversion
  - Mise à zéro des drapeaux par écriture de 1
  - Le bit SCF de ATDxSTAT0 indique la fin de séquence

```
; attente de fin de séquence
WTCONV:  BRCLR ATD0STAT0,#$80,WTCONV
```

- Fonctions des registres ATDxDIEN : un bit à 1 valide la broche PADx comme entrée/sortie numérique

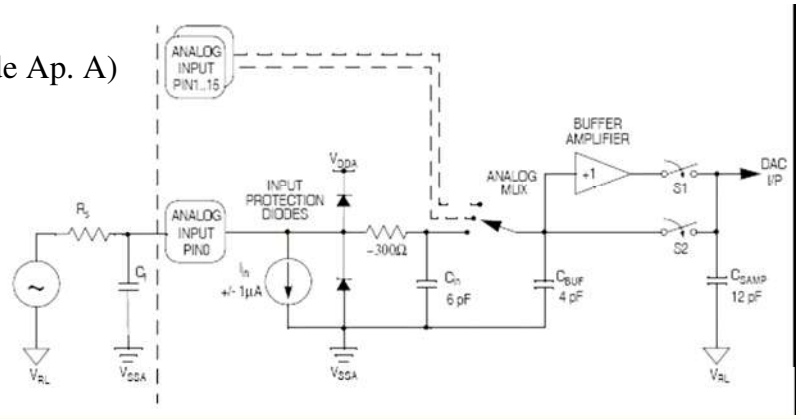
- PORTADx : registre de données du port numérique

4.6 Cas du HCS12

- Fonctions des registres de résultat de conversion :
  - ATDxDRLy et ATDxDRHy (x = 0 ou 1, y = 0 à 7) : résultat de conversion 10 bits

- Montage type pour une erreur d'échantillonnage < 1 LSB
  - Capacité de filtrage C_f > Nbits . (Cins – Cinn) (~ 3 nF pour 8 bits)
  - Résistance série R_s < 10 kΩ (pour 8 bits) ou 2,5 kΩ (pour 10 bits)

(Cf. Device_guide Ap. A)



4.6 Cas du HCS12

- Exemple d'une acquisition de 5 x 4 mesures sur entrée AN7
  - Horloge de CAN à 1 MHz
  - Horloge de bus à 2 MHz
    (fOSCCLK = 4 MHz, PLL non validé)

  - Programme principal

```
    INCLUDE 'mc9s12dp256.inc'
; data
    ORG $1200
DATA:     RMB 40
; code section
     ORG $1000
Entry:
    CLR COPCTL  ; clear watchdog
    LDS #$3FFE ; initialize stack pointer
    JSR OPENAD0
    JSR MEASURE
    BGND
```

4.6 Cas du HCS12

- Exemple d'une acquisition de 5 x 4 mesures sur entrée AN7

```
;*************************************************************
;*              Subroutines                                  *
;*************************************************************
OPENAD0: MOVB #$E0,ATD0CTL2
; Enable AD0 (set bit 7 ADPU to 1), Select fast flag clear all (set bit 6 AFFC to 1)
; Stop AD0 when wait mode (set bit 5 AWAI to 1), Disable external trigger on channel 7 (set bits 4, 3, and 2 to 0)
; Disable AD0 interrupt (set bit 1 ASCIF to 0)
              LDY #$C8   ; delay 100us
STALL:        DBNE Y,STALL
              MOVB #$22,ATD0CTL3
; Perform four conversions
; Disable FIFO mode
; When BDM becomes active, complete the current conversion then freeze
; Select 10-bit operation (set bit 7 SRES8 to 0)
               MOVB #$05,ATD0CTL4
; Two A/D clock periods for sample time (set bits 6 and 5 to 00)
; Set the value of PRS4~PRS0 to 00000 -> diviseur total de 2 de 2 MHz
              RTS
```


4.6 Cas du HCS12

- Exemple d'une acquisition de 5 x 4 mesures sur entrée AN7

```
;*************************************************************
;*              Subroutines                                  *
;*************************************************************
MEASURE:
;  5*4 measurements
              LDX #DATA ; initialize X
              LDY #5       ; initialize Y
LOOP5:        MOVB #$87,ATD0CTL5; start an A/D conversion sequence
; Result register right justified (set bit 7 DJM to 1)
; Result is unsigned (set bit 6 DSGN to 0)
; Nonscan mode (set bit 5 SCAN to 0)
; Single channel mode (set bit 4 MULT to 0)
; Select channel 7 (set bits 2..0 to 111)
              BRCLR ATD0STAT0,$80,*
              MOVW ATD0DR0,2,x+ ; collect and save the conversion results b6 à b15
               MOVW ATD0DR1,2,x+; post-increment the pointer by 2
               MOVW ATD0DR2,2,x+;
               MOVW ATD0DR3,2,x+;
              DBNE Y,LOOP5
              RTS
```

4.7 Tests

1. Donnez la gamme dynamique d'un convertisseur 12 bits.

2. Un signal analogique a un spectre de 200 à 3900 Hz. On a l'intention de l'échantillonner à 10 kHz. Ce choix est-il judicieux ?

3. Le convertisseur HCS12 a une tension de référence de 0 à 5 V. Donnez la résolution du convertisseur.

4. La sortie numérique d'un convertisseur a tous ses bits à un. Est-ce que cela correspond à la pleine échelle de l'entrée analogique ?

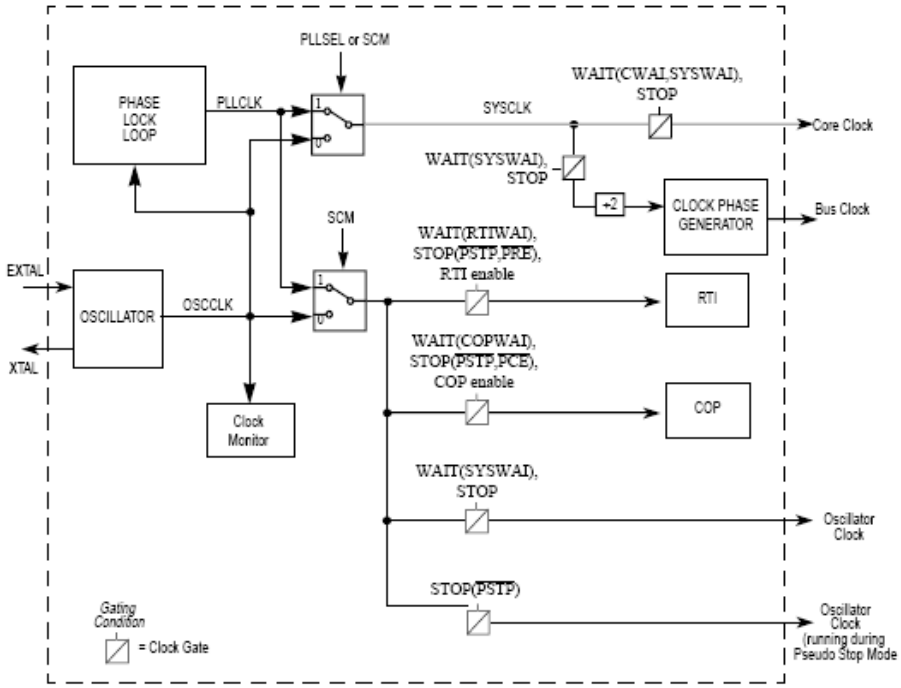
5. À quelle tension correspond la sortie 1011 d'un convertisseur [0-5V] ?

Chapitre 5 : Timer du HCS12

- Sommaire

5.1 Module d'horloge

- Générateur d'horloges



5.1 Module d'horloge

- Générateur d'horloges
  - choix de l'horloge du quartz externe (4 MHz), ou d'une fréquence multiple issue d'une boucle à verrouillage de phase (PLL)

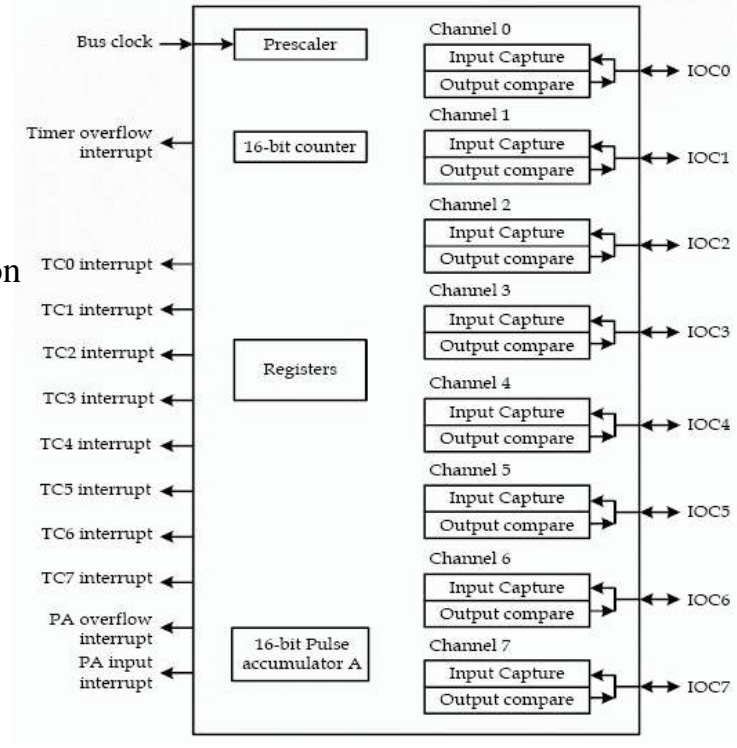
registre CLKSEL ($39)

	7	6	5	4	3	2	1	0
R W	PLLSEL	PSTP	SYSWAI	ROAWAI	PLLWAI	CWAI	RTIWAI	COPWAI
RESET:	0	0	0	0	0	0	0	0

  - utilisation de l'horloge de bus pour les modules timer (ECT) et conversion analogique-numérique (ATD)
  - génération de ECLK pour horloge externe et communication avec des mémoires externes

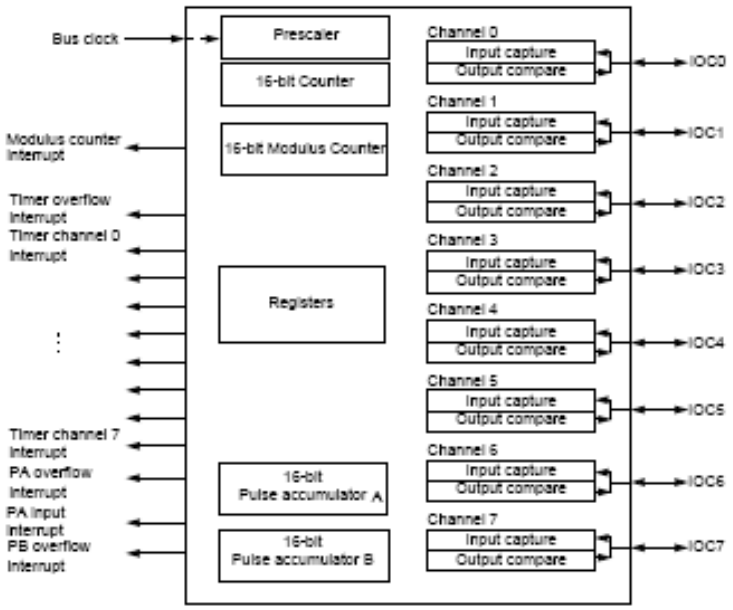
5.2 Module timer

- Le HCS12 a un module standard TIM qui comprend :
  - 8 canaux de capture d'entrée et de comparaison de sortie (port T)
  - 1 accumulateur pulsé 16 bits
  - 1 compteur 16 bits



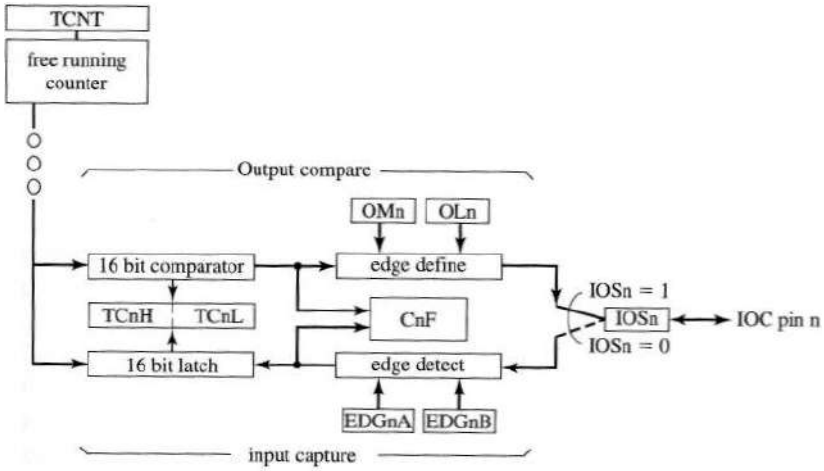
5.2 Module timer

- Le MC9S12DP256B comprend un module timer à capture renforcée (ECT) :
  - Caractéristiques principales du TIM
  - 1 tampon 16 bits pour chaque entrée de capture
  - 2 accumulateurs pulsés 16 bits
  - 1 décompteur 16 bits avec prédiviseur 4 bits
  - 4 compteurs de retard pour immuniser face au bruit



5.2 Module timer

- Le module TIM remplit 4 fonctions principales :
  - compteur libre 16 bits TCn
  - capture d'entrée IOCn
  - comparaison en sortie
  - accumulateur pulsé



5.3 Compteur libre

- Lorsque le timer est validé (bit TEN du registre TSCR1), le compteur compte de $0000 à $FFFF et se réinitialise automatiquement.
- La fréquence d'horloge est celle du bus divisée par 2^n avec n programmé sur 3 bits (bits PR0 à PR2 du registre TSCR2).
- Registres de contrôle TSCR1 ($ 46) et TSCR2 ($4D)

	7	6	5	4	3	2	1	0
	TEN	TSWAI	TSFRZ	TFFCA	0	0	0	0
value after reset	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
	TOI	0	0	0	TCRE	PR2	PR1	PR0
value after reset	0	0	0	0	0	0	0	0

5.3 Compteur libre

- Registres de comptage :
  registre 16 bits TCNT, concatenation de TCNTHi ($44) et TCNTLo ($45)

- Registre de drapeau de dépassement TFLG2

	BIT7	6	5	4	3	2	1	BIT0
R	TOF	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

passage de TOF à 1 à chaque dépassement du compteur (transition $FFFF à $0000)

5.4 Capture et comparaison de canaux

- Capture d'une entrée ou comparaison de sortie IOC0 à 7 (PT0 à PT7) selon le registre de sélection TIOS ($40)

  TIOSn = 0 : IOCn en entrée
  TIOSn = 1 : IOCn en sortie

	BIT7	6	5	4	3	2	1	BIT0
R W	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
RESET:	0	0	0	0	0	0	0	0

- Registre de drapeau d'évènement TFGL1 ($0E)

	BIT7	6	5	4	3	2	1	BIT0
R W	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
RESET.	0	0	0	0	0	0	0	0

5.4 Capture et comparaison de canaux

- Capture d'événement (front) sur une entrée
  - programmation du front actif par les registres de contrôle TCTL3 ($8A) et TCTL4 ($8B)

	7	6	5	4	3	2	1	0
	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
value after reset	0	0	0	0	0	0	0	0

(a) Timer control register 3 (TCTL3)

	7	6	5	4	3	2	1	0
	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
	0	0	0	0	0	0	0	0

(b) Timer control register 4 (TCTL4)

EDGnB EDGnA -- Edge configuration

0	0 : Capture disabled
0	1 : Capture on rising edges only
1	0 : Capture on falling edges only
1	1 : Capture on both edges

5.4 Capture et comparaison de canaux

- Exemple de mesure de période sur une entrée IC0

```
    INCLUDE 'mc9s12dp256.inc'
    ORG $1200
edge1 RMB 2; memory to hold the first edge
period RMB 2; memory to store the period
    ORG $1000
    MOVB #$90,TSCR1; enable timer counter and enable fast timer flags clear
    BCLR TIOS, mTIOS_IOS0; enable input-capture 0
    MOVB #$02,TSCR2; disable TCNT overflow interrupt, set prescaler to 4
    MOVB #$01,TCTL4; capture the rising edge of PT0 signal
    MOVB #mTFLG1_C0F,TFLG1; clear the C0F flag
    BRCLR TFLG1, mTFLG1_C0F,*; wait for the arrival of the first rising edge
    LDD TC0; save the first edge and clear the C0F flag
    STD edge1
    BRCLR TFLG1, mTFLG1_C0F,*; wait for the arrival of the second edge
    LDD TC0
    SUBD edge1; compute the period
    STD period
    BGND
    END
```


5.4 Capture et comparaison de canaux

- Le HCS12 a 8 fonctions de comparaison de sortie
- Chaque canal de comparaison de sortie comprend :
  - un comparateur 16 bits
  - un registre de comparaison TCn
  - une sortie active du port PTn
  - un circuit de réponse à interruption
  - une fonction de comparaison forcée
  - logique de contrôle
- Application à la génération de signal avec forme spécifiée (fréquence, rapport cyclique, modulation de largeur d'impulsions, …)

5.4 Capture et comparaison de canaux

- Lorsque le compteur TCNT a la même valeur que CTn, le drapeau CnF est mis à 1 ; la sortie PTn génère un évènement spécifié par les bits OMn/OLn des registres TCTL1 ($48) et TCTL2 ($49).

	7	6	5	4	3	2	1	0
	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
value after reset	0	0	0	0	0	0	0	0

(a) TCTL1 register

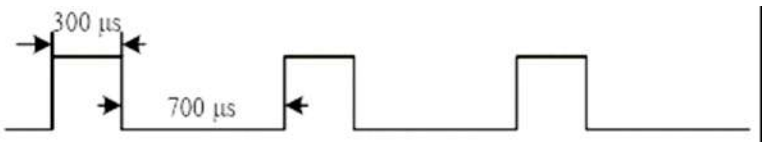
	7	6	5	4	3	2	1	0
	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
value after reset	0	0	0	0	0	0	0	0

(b) TCTL2 register
read: anytime
write: anytime

OMn	OLn	: output level
0	0	no action (timer disconnected from output pin)
0	1	toggle OCn pin
1	0	clear OCn pin to 0
1	1	set OCn pin to high

5.4 Capture et comparaison de canaux

- Exemple de génération de signal 1 kHz avec rapport cyclique 30 %



Si on considère que la fréquence du bus est de 2 MHz, on peut prendre un prédiviseur de 2. Le nombre de cycles d'horloge est de 300 pour le signal haut et de 700 pour le signal bas.

5.4 Capture et comparaison de canaux

- Programme de génération de signal 1 kHz sur OC0

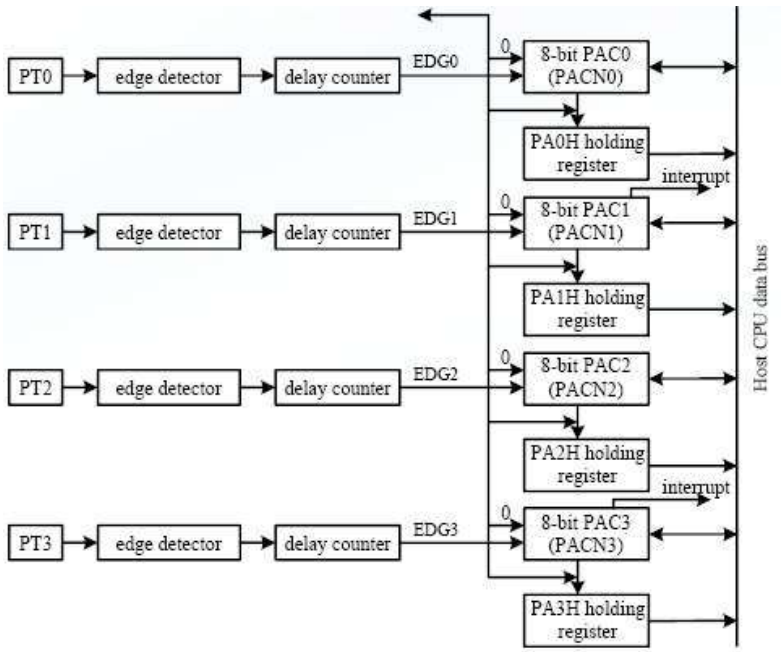
```
    INCLUDE 'mc9s12dp256.inc'
hi_time EQU 300
lo_time EQU 700
    ORG $1000
    MOVB #$90,TSCR1 ; enable TCNT with fast timer flag clear
    MOVB #$00,TSCR2 ; disable TCNT interrupt, set prescaler to 2
    bset TIOS, mTIOS_IOS0 ; enable OC0
    MOVB #$03,TCTL2 ; select pull high as pin action
    LDD TCNT ; start an OC0 operation with 700 us as delay
repeat: ADDD #lo_time ; "
    STD TC0 ; "
low:   BRCLR TFLG1, mTFLG1_C0F,low ; wait until OC0 pin go high
    MOVB #$02,TCTL2 ; select pull low as pin action
    LDD TC0 ; start an OC operation with 300 us as delay
    ADDD #hi_time ; "
    STD TC0 ; "
high: BRCLR TFLG1, mTFLG1_C0F,high ; wait until OC0 pin go low
    MOVB #$03,TCTL2 ; select pull high as pin action
    LDD TC0
    BRA repeat
```


5.5 Accumulateur pulsé

- Le HCS12 ECT possède 4 accumulateurs pulsés 8 bits (PAC0 à PAC3)
- Deux accumulateurs peuvent être concaténés en un accumulateur pulsé 16 bits
- Ils existent 4 configurations possibles :
  - 2 accumulateurs pulsés 16 bits PACA et PACB
  - 1 accumulateur 16 bits PACA et 2 accumulateurs 8 bits PAC0 et PAC1
  - 4 accumulateurs pulsés 8 bits PAC0 à PAC3
  - 4 accumulateurs pulsés 8 bits PAC0 à PAC3 avec sorties sur PT0 à PT3
- Lorsque les accumulateurs 16 bits sont concaténés, PACA a une sortie sur PT7 et PACB une sortie sur PT0.

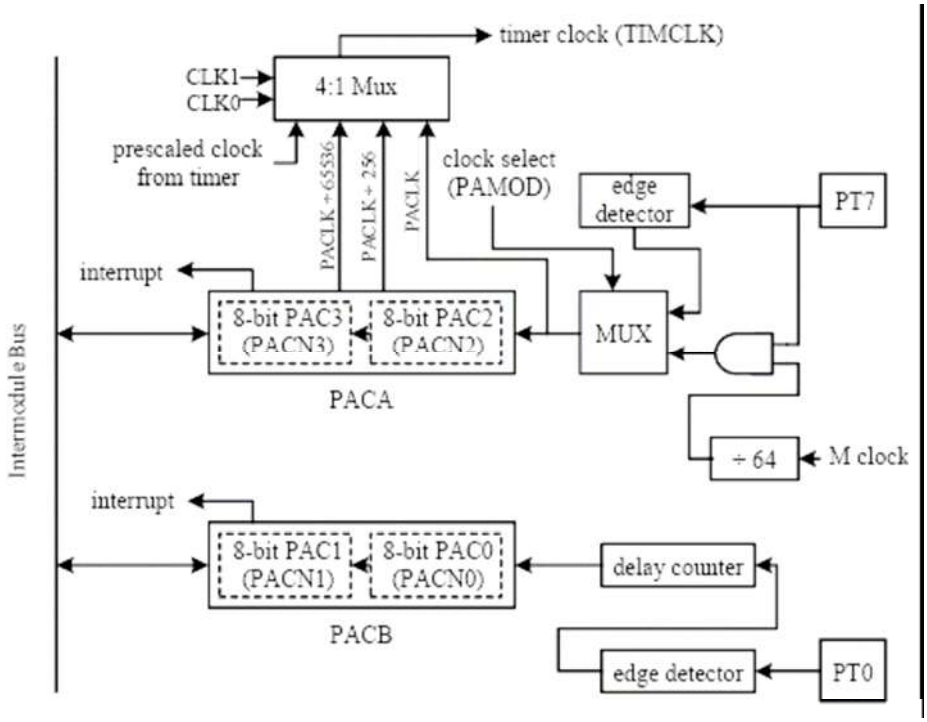
5.5 Accumulateur pulsé

- Accumulateur pulsé 8 bits
  compteur d'impulsions sur une entrée



5.5 Accumulateur pulsé

- Accumulateur pulsé 16 bits



5.5 Accumulateur pulsé

- 2 modes d'opération :
  - Mode compteur d'évènements

    L'accumulateur PACA peut opérer dans ce mode et compter le nombre d'évènements sur l'entrée PT7. L'accumulateur PACB et les 4 accumulateurs 8 bits n'opèrent que dans ce mode.
  - Mode accumulateur commandé

    L'accumulateur PACA peut opérer dans ce mode. Tant que l'entrée PT7 est active, le compteur PACA est synchronisé par une horloge du bus prédivisée par 64.

- Le front actif de PACB et de PAC0 à PAC3 agit de manière similaire à celui de IC0 à IC3 vu précédemment. Il est nécessaire de spécifier le type de front actif dans le registre TCTL4.

5.5 Accumulateur pulsé

- 2 sources d'interruption pour PACA :

  entrée PT7 et dépassement du compteur PACA

- génération possible d'interruption pour les accumulateurs 8 bits PAC1 et PAC3 (dépassement du compteur)

- génération possible d'interruption pour l'accumulateur 16 bits PACB (dépassement du compteur haut 8 bits)

5.5 Accumulateur pulsé

- L'accumulateur PACA est associé à des registres de contrôle PACTL ($60), de drapeau d'état PAFLG ($61) et de compteur 16 bits PACN32 ($62); l'accumulateur PACB est également associé à PBCTL ($70), PBFLG ($71).

	7	6	5	4	3	2	1	0
	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
reset:	0	0	0	0	0	0	0	0

Pulse accumulator control register (PACTL)

	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	PAOVF	PAIF
reset:	0	0	0	0	0	0	0	0

Pulse accumulator flag register (PAFLG)

5.5 Accumulateur pulsé

- Chaque accumulateur 8 bits peut être validé par programmation du registre ICPAR ($68). Il a aussi un registre de maintien PA0H ($75) à PA3H ($72).
- L'utilisateur peut éviter le dépassement des accumulateurs 8 bits en spécifiant le bit PACMX du registre ICSYS ($70).

	7	6	5	4	3	2	1	0
	0	0	0	0	PA3EN	PA2EN	PA1EN	PA0EN
reset:	0	0	0	0	0	0	0	0

Input control pulse accumulator control register (ICPACR)

	7	6	5	4	3	2	1	0
	SH37	SH26	SH15	SH04	TFMOD	PACMX	BUFEN	LATQ
reset:	0	0	0	0	0	0	0	0

Input control system control register (ICSYS)

5.5 Accumulateur pulsé

- Exemple de comptage de N impulsions sur PT7 avec interruption

```
        INCLUDE 'mc9s12dp256.inc'
N EQU 1350;
    ORG $1000
    LDS #$3FFF; set up stack pointer
    MOVW #paov_isr, Vtimpabovf ; set up PAOV interrupt vector
    LDD #N; place the 2s complement in PACNT
    COMA;“
    COMB;“
    ADDD #1;“
    STD PACN32;“
    MOVB #$52,PACTL ; enable PACA, event counting mode, active
; edge is rising
    CLI; enable PAOV interrupt
Loop: JMP Loop
    BGND
paov_isr:  MOVB #mPAFLG_PAOVF,PAFLG; clear the PAOVF flag
    RTI
    END
```


5.6 Applications

- Modulation de largeur d'impulsion pour commande de moteurs continus (rapport cyclique variable)
- Générateur d'alarmes
- Contrôle d'accès (digicode)
- Compteur de vitesse (anémomètre)
- Mesures de temps (détecteur IR)
- Générateur de signaux

5.7 Tests

1. Dans l'exemple du programme de mesure de période, trouvez la période maximale qui ne provoque pas de dépassement du compteur ; on supposera la fréquence du bus égale à 2 MHz.

2. Si le compteur a été capturé à $0037 et à $FB20, calculez la période en secondes.

3. Proposez un programme pour la mesure de rapport cyclique.

4. Comment doit-on mettre à zéro un drapeau CFn ?

5. Décrivez les 2 modes d'opération de l'accumulateur pulsé PACA.

6. Proposez un programme utilisant l'accumulateur pulsé PACA pour la mesure de période.

Chapitre 6 : Timer avancé du HCS12

- Sommaire

6.1 Module RTI

- Interruption temps réel (Real-Time Interrupt)
  - Générateur d'horloges (CRG) avec horloge de base OSCCLK
  - Vecteur d'interruption Vrti ($FFF0-$FFF1)
  - Registre de validation CRGINT ($38) : bit de validation RTIE

	7	6	5	4	3	2	1	0
R	RTIE	0	0	LOCKIE	0	0	SCMIE	0
W								
RESET:	0	0	0	0	0	0	0	0

  - Registre d'état CRGFLG ($37) : bit de drapeau RTIF

	7	6	5	4	3	2	1	0
R	RTIF	PORF	0	LOCKIF	LOCK	TRACK	SCMIF	SCM
W								
RESET:	0	!	0	0	0	0	0	0

  - Registre de contrôle RTICTL ($3B)

	7	6	5	4	3	2	1	0
R	0	RTR6	RTR5	RTR4	RTR3	RTR2	RTR1	RTR0
W								
RESET:	0	0	0	0	0	0	0	0

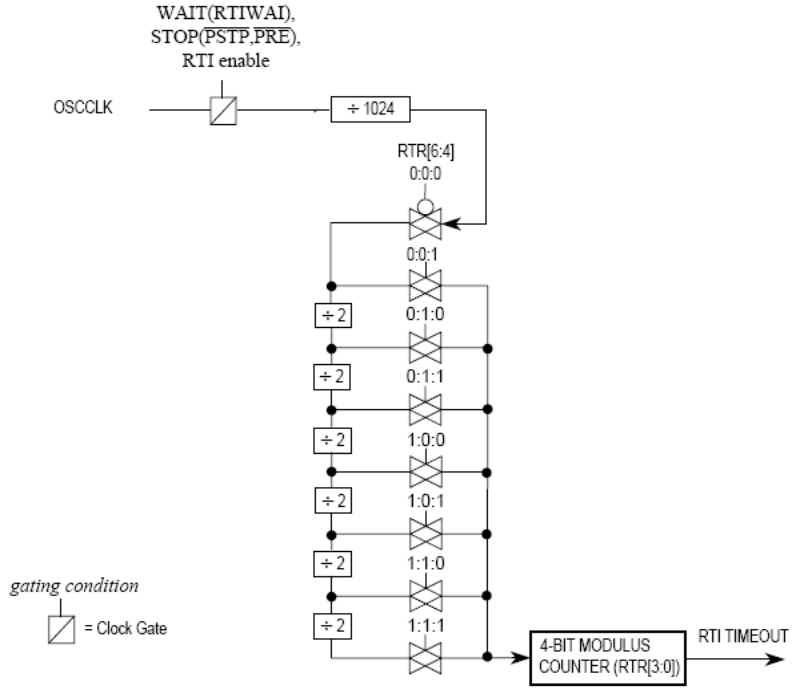
6.1 Module RTI

- Interruption temps réel (Real-Time Interrupt)
  - Registre de contrôle RTICTL ($3B)
    - Prédiviseur programmable sur 7 bits RTR0 à RTR6 pour obtenir des prédiviseurs de 2^{10} à 2^{17}

RTR[3:0]	RTR[6:4]							
	000 (off)	001 (2^{10})	010 (2^{11})	011 (2^{12})	100 (2^{13})	101 (2^{14})	110 (2^{15})	111 (2^{16})
0000 (÷1)	off*	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
0001(÷2)	off*	2×2^{10}	2×2^{11}	2×2^{12}	2×2^{13}	2×2^{14}	2×2^{15}	2×2^{16}
0010 (÷3)	off*	3×2^{10}	3×2^{11}	3×2^{12}	3×2^{13}	3×2^{14}	3×2^{15}	3×2^{16}
0011 (÷4)	off*	4×2^{10}	4×2^{11}	4×2^{12}	4×2^{13}	4×2^{14}	4×2^{15}	4×2^{16}
0100 (÷5)	off*	5×2^{10}	5×2^{11}	5×2^{12}	5×2^{13}	5×2^{14}	5×2^{15}	5×2^{16}
0101 (÷6)	off*	6×2^{10}	6×2^{11}	6×2^{12}	6×2^{13}	6×2^{14}	6×2^{15}	6×2^{16}
0110 (÷7)	off*	7×2^{10}	7×2^{11}	7×2^{12}	7×2^{13}	7×2^{14}	7×2^{15}	7×2^{16}
0111 (÷8)	off*	8×2^{10}	8×2^{11}	8×2^{12}	8×2^{13}	8×2^{14}	8×2^{15}	8×2^{16}
1000 (÷9)	off*	9×2^{10}	9×2^{11}	9×2^{12}	9×2^{13}	9×2^{14}	9×2^{15}	9×2^{16}
1001 (÷10)	off*	10×2^{10}	10×2^{11}	10×2^{12}	10×2^{13}	10×2^{14}	10×2^{15}	10×2^{16}
1010 (÷11)	off*	11×2^{10}	11×2^{11}	11×2^{12}	11×2^{13}	11×2^{14}	11×2^{15}	11×2^{16}
1011 (÷12)	off*	12×2^{10}	12×2^{11}	12×2^{12}	12×2^{13}	12×2^{14}	12×2^{15}	12×2^{16}
1100 (÷13)	off*	13×2^{10}	13×2^{11}	13×2^{12}	13×2^{13}	13×2^{14}	13×2^{15}	13×2^{16}
1101 (÷14)	off*	14×2^{10}	14×2^{11}	14×2^{12}	14×2^{13}	14×2^{14}	14×2^{15}	14×2^{16}
1110 (÷15)	off*	15×2^{10}	15×2^{11}	15×2^{12}	15×2^{13}	15×2^{14}	15×2^{15}	15×2^{16}
1111 (÷16)	off*	16×2^{10}	16×2^{11}	16×2^{12}	16×2^{13}	16×2^{14}	16×2^{15}	16×2^{16}

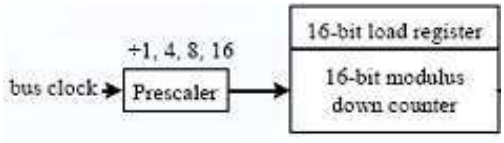
6.1 Module RTI

- Fonctionnement
  - Drapeau RTIF à 1 à chaque période, mise à zéro par écriture de 1



6.2 Module décompteur

- Module décompteur (Modulus Down-Counter)
  - Générateur d'interruptions périodiques
  - Utilisation pour mise en mémoire des registres IC (Input Capture) et de l'accumulateur pulsé PA
  - Mise en mémoire unique ou périodique
  - Horloge de base : bus CLOCK



  - Registre de contrôle MCCTL ($66)
    - Prédiviseur programmable à 1, 4, 8 et 16 (MCPR0 et MCPR1)

	BIT7	6	5	4	3	2	1	BIT0
R	MCZI	MODMC	RDMCL	0	0	MCEN	MCPR1	MCPR0
W				ICLAT	FLMC			
RESET:	0	0	0	0	0	0	0	0

6.2 Module décompteur

- Module décompteur (Modulus Down-Counter)
  - Registre d'état MCFLG ($67)
    - Drapeau de passage à zéro MCZF, état de polarité d'entrée IC

	BIT7	6	5	4	3	2	1	BIT0
R	MCZF	0	0	0	POLF3	POLF2	POLF1	POLF0
W								
RESET:	0	0	0	0	0	0	0	0

  - Registre de décomptage 16 bits MCCNT ($76)
    - Registre de chargement utilisé après chaque passage à zéro du décompteur
    - Écriture double dans le décompteur et dans le registre de chargement

6.2 Module décompteur

- Exemple : sous-programme générant un retard multiple de 10 ms

```
; Y contains the multiple of 10 ms
; bus clock is 2 MHz
Delay10ms: BSET TSCR1,mTSCR1_TFFCA; enable timer fast flag clear
    MOVB #$05,MCCTL; enable modulus down counter (MCEN = 1)
                   ; with 1/4 as prescaler (MCPR1 = 0, MCPR0 = 1)
    MOVW #5000,MCCNT; load the value to be down counted
    BRCLR MCFLG,mMCFLG_MCZF,*
    BCLR MCCTL,$04; disable modulus down counter (MCEN = 0)
    DBNE Y,Delay10ms
    RTS
```


6.3 Timer ECT

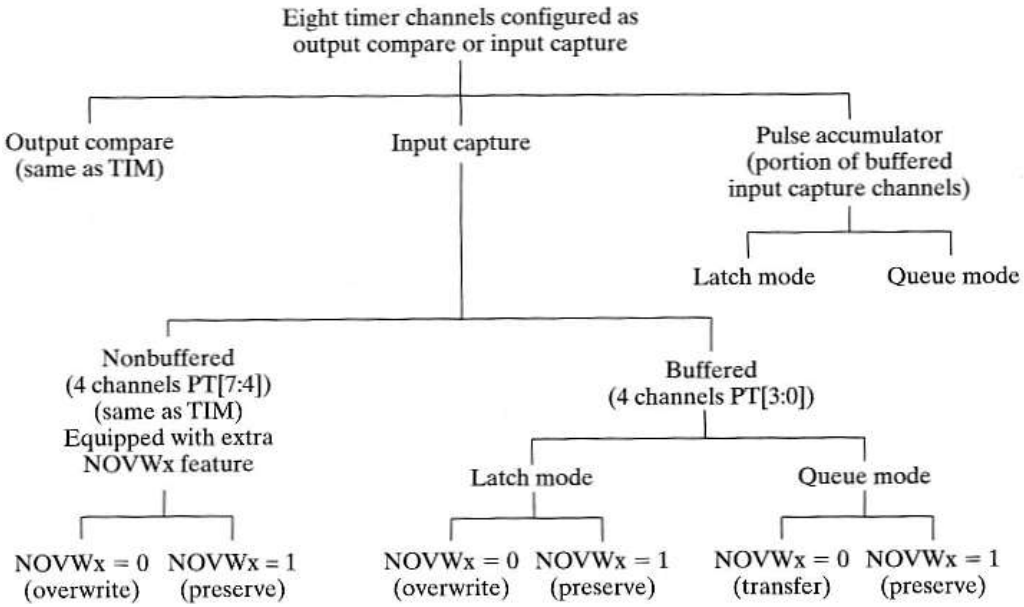
- Module Timer à capture renforcée (Enhanced Capture Timer)
  - Caractéristiques principales du module TIM (chapitre précédent)
  - 4 accumulateurs pulsés 8 bits
  - 1 décompteur 16 bits
  - 4 compteurs de retard pour une meilleure immunité au bruit
  - Options automatisant les mesures

  - Registre de maintien 16 bits pour chaque entrée IC0 à IC3
    - Option de non effacement de la dernière valeur (NoOverWrite) dans le registre ICOVW ($6A) avant lecture
    - Registre vidé après lecture

	BIT7	6	5	4	3	2	1	BIT0
R W	NOVW7	NOVW6	NOVW5	NOVW4	NOVW3	NOVW2	NOVW1	NOVW0
RESET:	0	0	0	0	0	0	0	0

6.3 Timer ECT

- Module Timer à capture renforcée (Enhanced Capture Timer)



6.3 Timer ECT

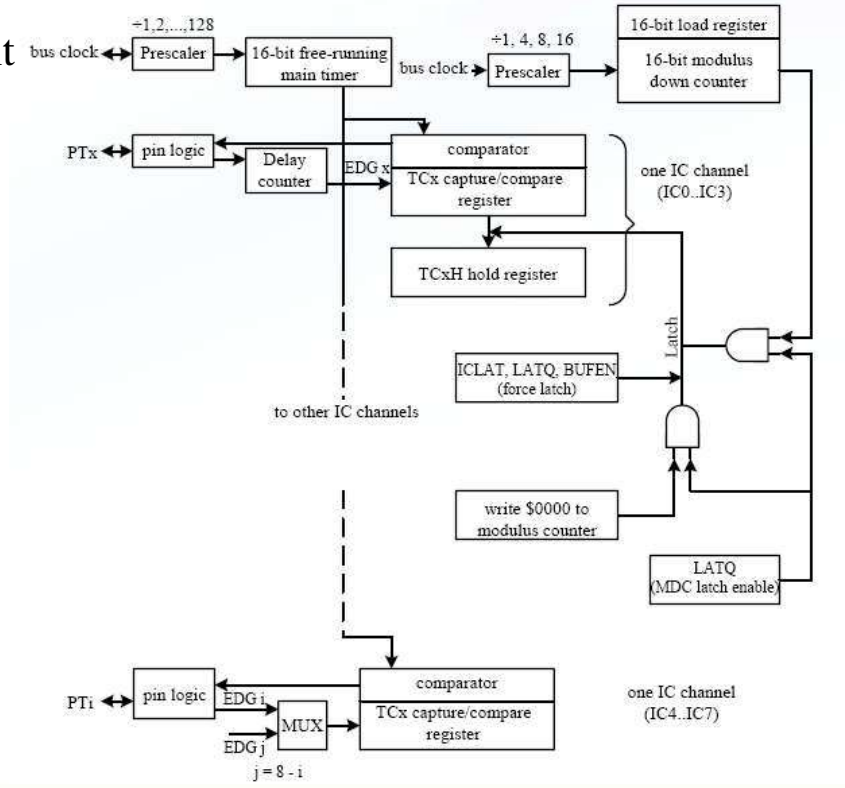
- Fonctionnement
  - Registre ICSYS ($6B) : bits de validation BUFEN et de mode LATQ

	BIT7	6	5	4	3	2	1	BIT0
R W	SH37	SH26	SH15	SH04	TFMOD	PACMX	BUFEN	LATQ
RESET:	0	0	0	0	0	0	0	0

  - Mode mise en mémoire (latch)
    - Mise en mémoire lorsque le décompteur atteint la valeur zéro ou lorsqu'on écrit dans le compteur
    - Au moment de la mise en mémoire, le contenu des registres IC et accumulateurs pulsés 8 bits est transféré dans les registres de maintien. Après l'opération, les accumulateurs pulsés sont vides.
  - Mode file d'attente (queue)
    - À l'arrivée d'un front actif de l'entrée x, la valeur principale du timer est copiée dans le registre TCx.
    - À l'arrivée d'un deuxième front, le contenu du registre TCx est transféré dans le registre de maintien.

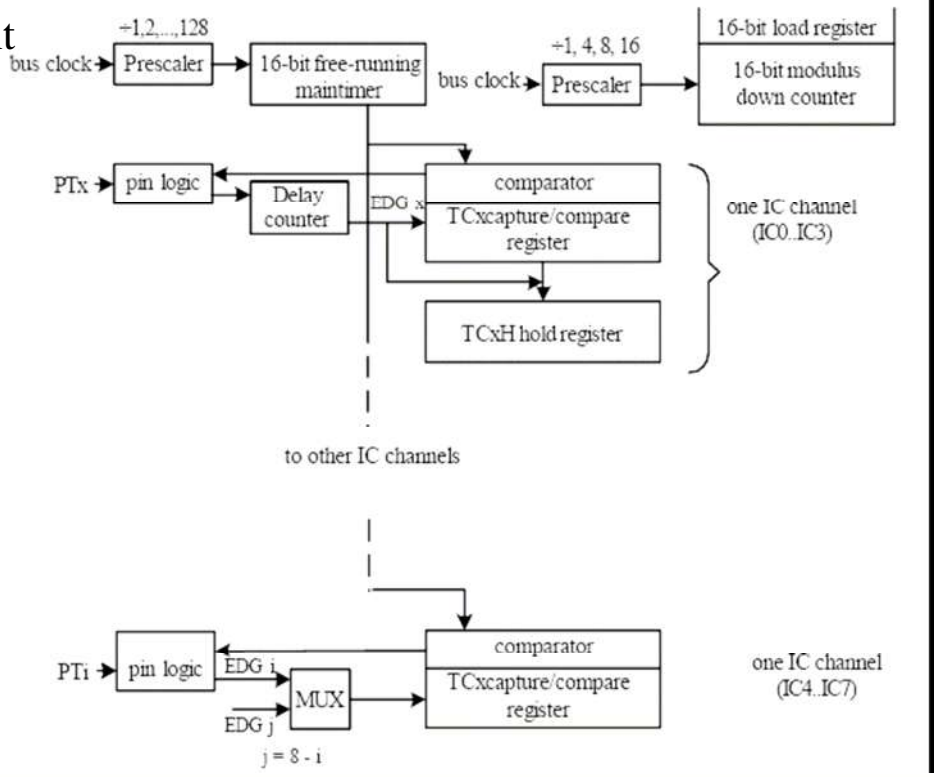
6.3 Timer ECT

- Fonctionnement IC
  mode mémoire
  (latch)



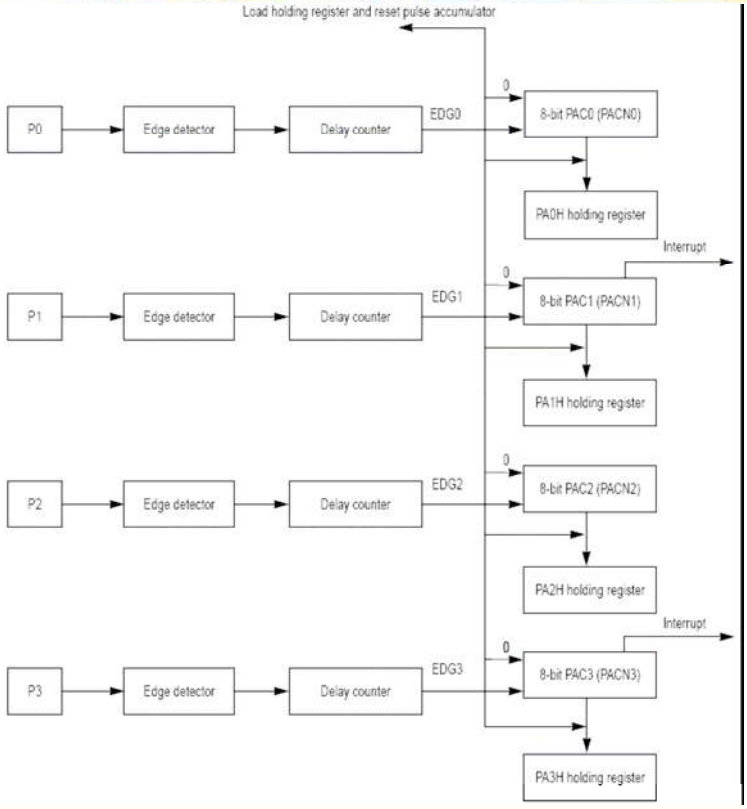
6.3 Timer ECT

- Fonctionnement IC
  mode
  file d'attente
  (queue)



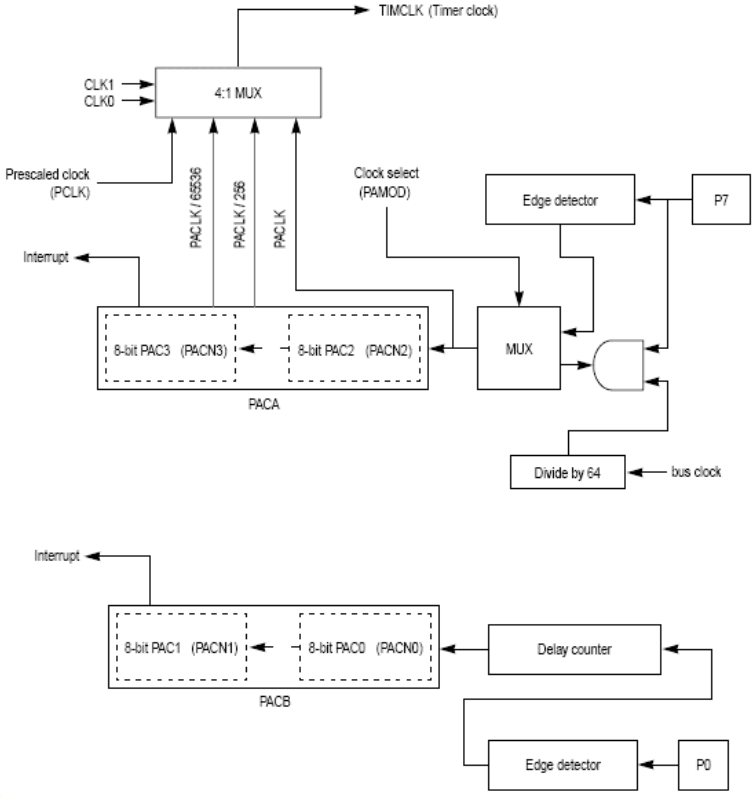
6.3 Timer ECT

- Fonctionnement PA 8 bits mode mémoire (latch)



6.3 Timer ECT

- Fonctionnement PA 16 bits mode mémoire (latch)



6.3 Timer ECT

- Compteur de retard pour distinguer les vrais fronts actifs de ceux dus au bruit
  - Le front est considéré comme vrai s'il dure plus qu'un temps préfixé de retard (delay).
  - La durée du retard est fixée par les bits DLY0 et DLY1 du registre DLYCT ($69).

	BIT7	6	5	4	3	2	1	BIT0
R	0	0	0	0	0	0	DLY1	DLY0
W								
RESET:	0	0	0	0	0	0	0	0

DLY1	DLY0	Delay
0	0	Disabled (bypassed)
0	1	256 bus clock cycles
1	0	512 bus clock cycles
1	1	1024 bus clock cycles

6.3 Timer ECT

- Exemple de mesure de durée d'impulsion sur PT0 en environnement bruyant (durée de bruit plus petit que 256 périodes d'horloge de bus)

```
            INCLUDE 'mc9s12dp256.inc'
            ORG $1500
edge1 RMB 2
overflow RMB 2
pulse_width RMB 2
            ORG $1000
            MOVW #tov_isr, Vtimpabovf; set up timer overflow interrupt vector
            LDD #0
            STD overflow
            MOVB #$90,TSCR1; enable TCNT and fast timer flag clear
            MOVB #$04,TSCR2; set prescaler to TCNT to 16
            BCLR TIOS, mTIOS_IOS0; enable input-capture 0
            MOVB #$01,DLYCT; set delay count to 256 bus cycles
            MOVB #$01,ICOVW; prohibit overwrite to TC0 register
            MOVB #$0,ICSYS; disable queue mode
            MOVB #$01,TCTL4; capture the rising edge on PT0 pin
```

6.3 Timer ECT

- Exemple de mesure de durée d'impulsion sur PT0 en environnement bruyant (durée de bruit plus petit que 256 périodes d'horloge de bus)

```
  MOVB #mTFLG1_C0F,TFLG1; clear C0F flag
; wait for arrival of the first rising edge
  BRCLR TFLG1, mTFLG1_C0F,*
  MOVB #mTFLG2_TOF,TFLG2; clear the TOF flag
; enable TCNT overflow interrupt
  BSET TSCR2, mTSCR2_TOI
  CLI; valid interrupt
; clear C0F and save the captured first edge
  MOVW TC0,edge1
; capture the falling edge on PT0 pin
  MOVB #$02,TCTL4
; wait for the arrival of the falling edge
  BRCLR TFLG1, mTFLG1_C0F,*
  LDD TC0
  SUBD edge1
  STD pulse_width
  BCC NEXT
; second edge is smaller,
; so decrement overflow count by 1
  LDX overflow
  DEX
  STX overflow
NEXT: BGND
tov_isr: MOVB #mTFLG2_TOF,TFLG2
; clear TOF flag
  LDX overflow; inc. TCNT overflow count
  INX
  STX overflow
  RTI
  END
```


6.4 Tests

1. Quelles sont les fonctions du décompteur ?

2. Si un compteur 16 bits incrémente toutes les 2 µs, quelle est la durée des dépassements réguliers ?

3. Proposez un programme pour la mesure d'une période sur PT0.

4. Quelles améliorations par rapport à HCS12 standard apportent le module ECT ?

5. Pour quel type d'application utilise-t-on le module RTI ?

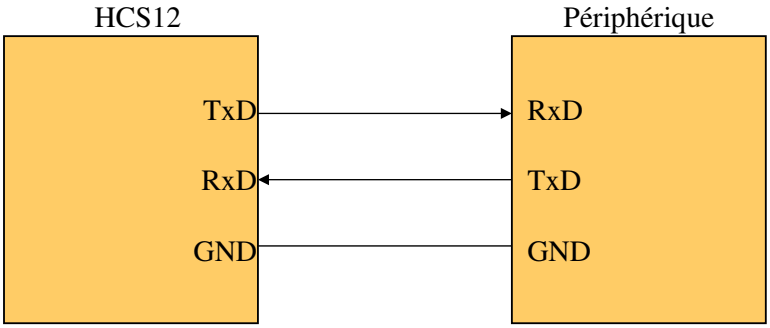
6. Proposez un programme utilisant le module RTI qui met à jour l'heure (hh:mm:ss) toutes les 8,196 ms (on suppose disposer d'une horloge OSCCLK à 2 MHz).

Chapitre 7 : Interface série asynchrone

- Sommaire

7.1 Interface série asynchrone

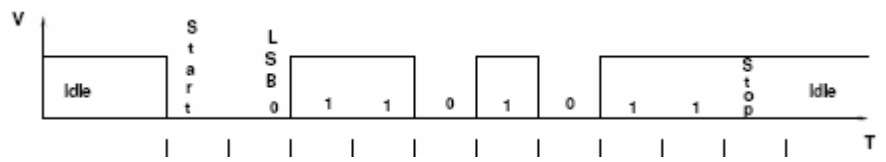
- Caractéristiques
  - Pas de ligne d'horloge, utilisation d'horloge interne de même fréquence
  - Transmission des données par série de bits (8 ou 9), avec bit de start, bit de parité et bit de stop
  - Les bits de start et de stop servent à la synchronisation du transfert
  - Ligne de transmission entre émetteur (TxD) et récepteur (RxD)



communication full-duplex (émission et réception simultanées)

7.1 Interface série asynchrone

- Fonctionnement
  - Exemple de transmission d'une donnée %11010110 au format NRZ (non retour à zéro)



  - Envoi des bits successifs LSB à MSB
  - Largeur d'un bit : inverse de la vitesse de transmission (Baud ou bit/s)
  - Ligne inactive (idle)
  - Synchronisation par les données
    - caractère 7 bits ASCII (American Standard Code Information Interchange)

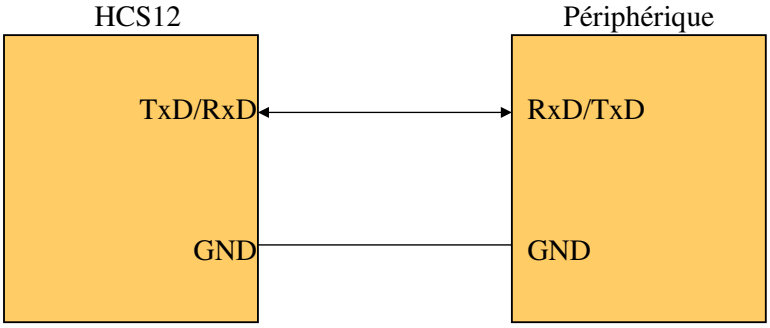
7.1 Interface série asynchrone

- Transmission ASCII 7 bits

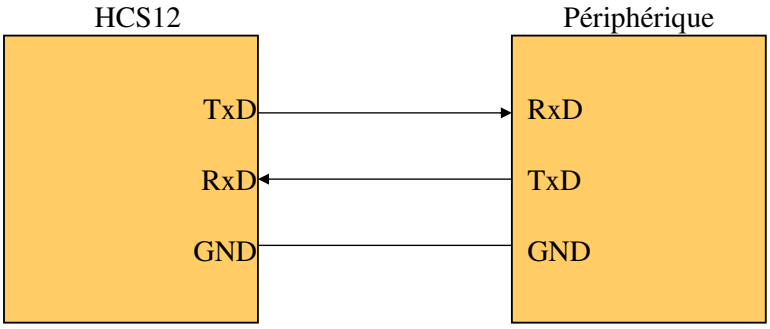
Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII
$00	NUL	$20	SP *space*	$40	@	$60	` *grave*
$01	SOH	$21	!	$41	A	$61	a
$02	STX	$22	" quote	$42	B	$62	b
$03	ETX	$23	#	$43	C	$63	c
$04	EOT	$24	$	$44	D	$64	d
$05	ENQ	$25	%	$45	E	$65	e
$06	ACK	$26	&	$46	F	$66	f
$07	BEL *beep*	$27	' *apost.*	$47	G	$67	g
$08	BS *back sp*	$28	(	$48	H	$68	h
$09	HT *tab*	$29	)	$49	I	$69	i
$0A	LF *linefeed*	$2A	*	$4A	J	$6A	j
$0B	VT	$2B	+	$4B	K	$6B	k
$0C	FF	$2C	, *comma*	$4C	L	$6C	l
$0D	CR *return*	$2D	- *dash*	$4D	M	$6D	m
$0E	SO	$2E	. *period*	$4E	N	$6E	n
$0F	SI	$2F	/	$4F	O	$6F	o
$10	DLE	$30	0	$50	P	$70	p
$11	DC1	$31	1	$51	Q	$71	q
$12	DC2	$32	2	$52	R	$72	r
$13	DC3	$33	3	$53	S	$73	s
$14	DC4	$34	4	$54	T	$74	t
$15	NAK	$35	5	$55	U	$75	u
$16	SYN	$36	6	$56	V	$76	v
$17	ETB	$37	7	$57	W	$77	w
$18	CAN	$38	8	$58	X	$78	x
$19	EM	$39	9	$59	Y	$79	y
$1A	SUB	$3A	:	$5A	Z	$7A	z
$1B	ESCAPE	$3B	;	$5B	[	$7B	{
$1C	FS	$3C	<	$5C	\	$7C	|
$1D	GS	$3D	=	$5D	]	$7D	}
$1E	RS	$3E	>	$5E	^	$7E	~
$1F	US	$3F	?	$5F	_ *under*	$7F	DEL *delete*

7.1 Interface série asynchrone

- Type de transmission



Communication half-duplex (émission et réception en deux temps)



Communication full-duplex (émission et réception simultanées)

7.1 Interface série asynchrone

- Parité
  - ➢ Détection d'une erreur de transmission
    - Calcul du bit de parité de la donnée
    - Ajout du bit à la fin de la donnée émise
    - Calcul du bit de parité de la donnée à la réception
    - Si bit de parité calculé à la réception différente de celle lue à la suite de la donnée, détection d'une erreur de transmission
  - ➢ Parité paire (even) : nombre de 1 pair (donnée + bit de parité)
  - ➢ Parité impaire (odd) : nombre de 1 impair (donnée + bit de parité)

7.1 Interface série asynchrone

- Parité
  - Exemple simplifié d'une parité paire pour un message 3 bits ($m_1 m_2 m_3$)
    - Le bit de contrôle k qui est ajouté à l'émission est en fait :

m_1	m_2	m_3	k
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$k = m_1 \otimes m_2 \otimes m_3$$

- Le test de parité à la réception génère un bit T tel que :

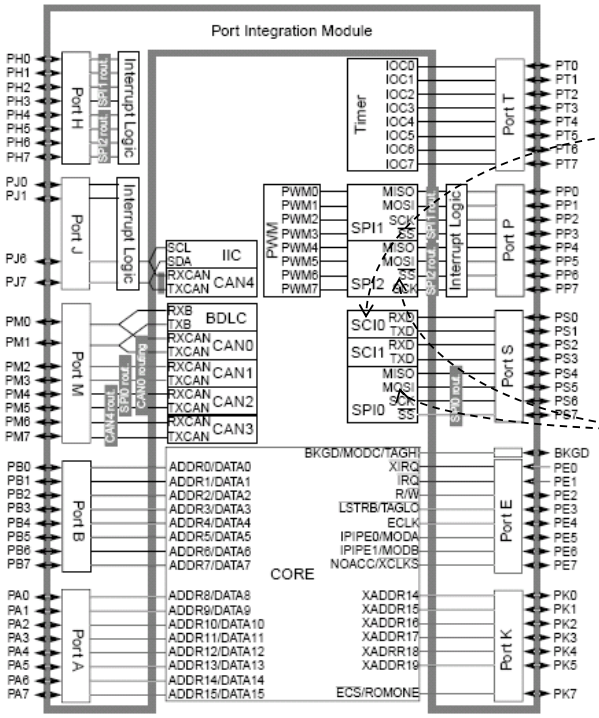
$$T = k' \otimes k = m_1 \otimes m_2 \otimes m_3 \otimes k$$

si T = 1 il y a erreur de transmission du message.

Cette détection d'erreur ne permet pas de corriger l'erreur (cf. code de Hamming).

7.1 Interface série asynchrone

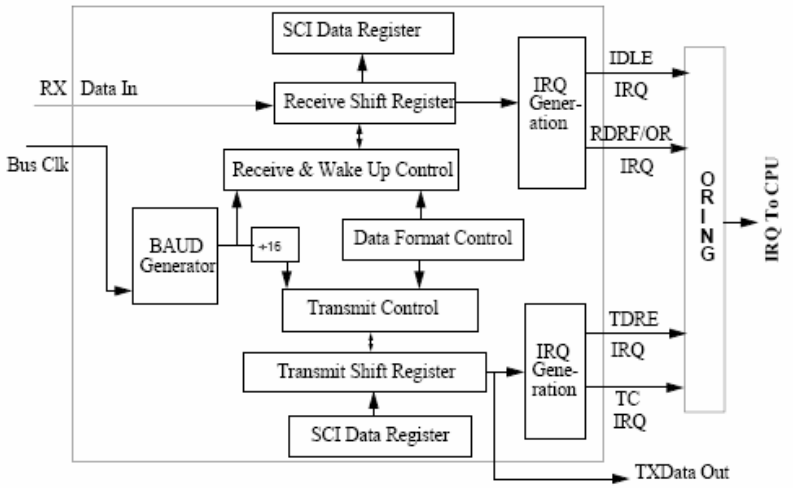
- 5 interfaces séries indépendantes pour le HCS12 :



- 2 interfaces série asynchrones (Serial Communication) :
  - SCI0, SCI1 (port S)
  - compatibles RS232
- 3 interfaces série synchrones (Serial Peripheral) :
  - SPI0 (port S), SPI1et SPI2 (port P)
  - horloge
  - communications avec périphériques et microcontrôleurs

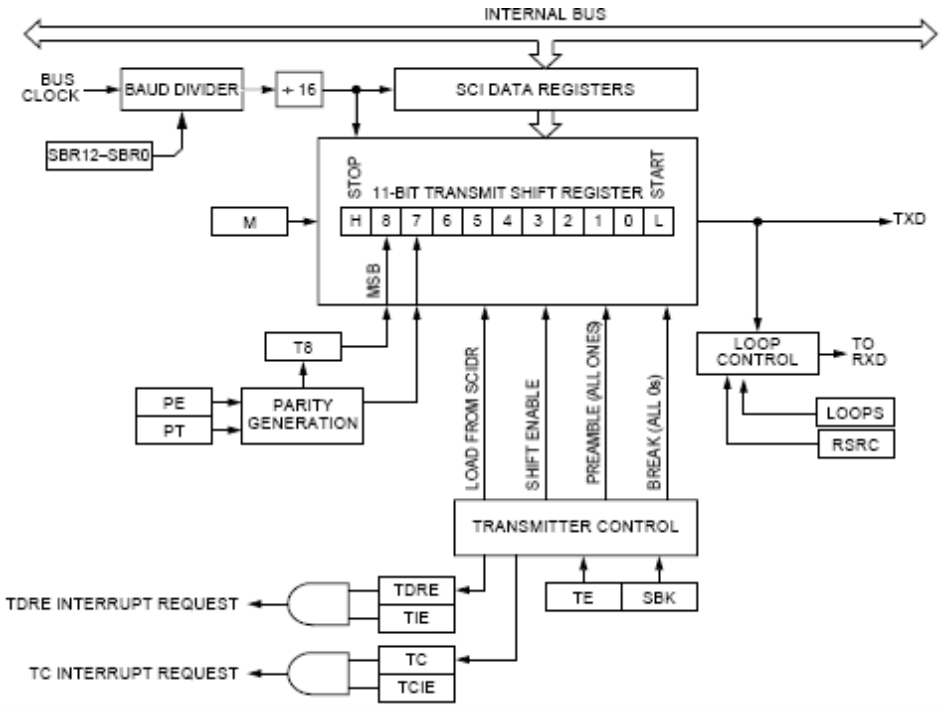
7.2 Module SCI

- Architecture du SCI (Serial Communication Interface)
  - Générateur d'horloge sur la base de Bus Clock
  - Registres à décalage à l'émission et à la réception
  - Réveil du récepteur par ligne inactive ou adressage
  - Détection de parité



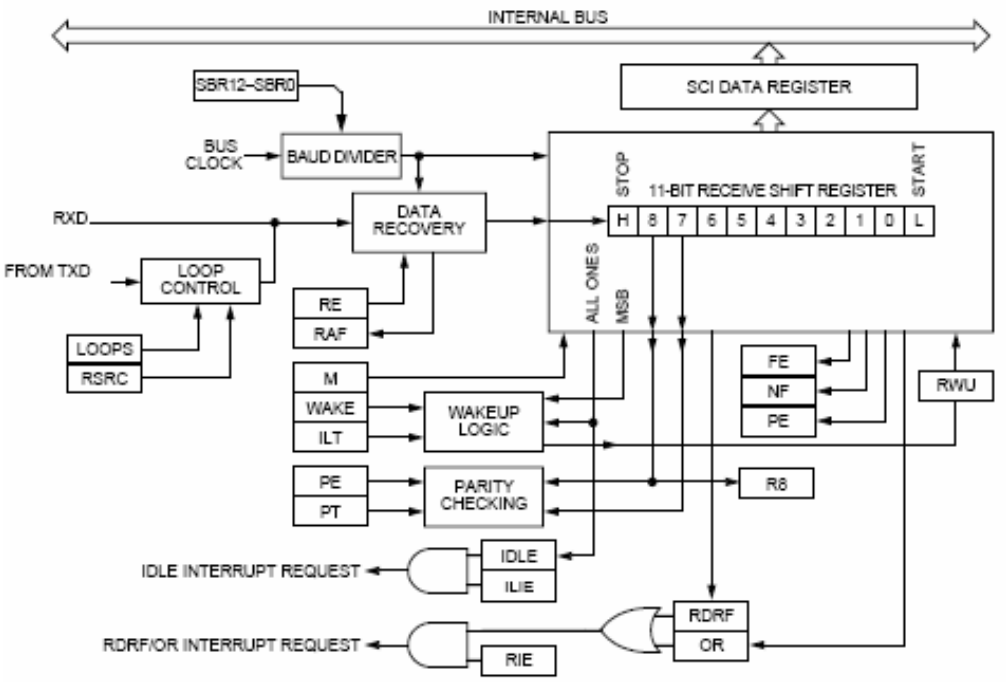
7.2 Module SCI

- Architecture de l'émetteur



7.2 Module SCI

- Architecture du récepteur



7.2 Module SCI

- Registres pour SCI0 ($C8-$CF)
  - Registres de vitesse de transmission (SCI0BDH et SCI0BDL)
  - Registres de contrôle (SCI0CR1 et SCI0CR2)
  - Registres d'état (SCI0SR1 et SCI0SR2)
  - Registres de données (SCI0DRH et SCI0DRL)

Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCI0BDH	Read:	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
	Write:								
SCI0BDL	Read:	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
	Write:								
SCI0CR1	Read:	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
	Write:								
SCI0CR2	Read:	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
	Write:								
SCI0SR1	Read:	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
	Write:								
SCI0SR2	Read:	0	0	0	0	0	BRK13	TXDIR	RAF
	Write:								
SCI0DRH	Read:	R8	T8	0	0	0	0	0	0
	Write:								
SCI0DRL	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0

7.2 Module SCI

- **SCIx BauD rate registers** :
  - Vitesse identique à l'émission et à la réception
  - Vitesse transmission = Bus CLK / (16 x SBR)
    où SBR : valeur sur 13 bits contenue dans SCIxBDH et SCIxBDL

ex.: SBR = 13 ; Bus CLK = 2 MHz ; Vitesse = 9600 Baud

- **SCIx Control Register 1** :
  - Longueur données de 8 ou 9 bits (M)
  - Parité : On/Off (TE), Type pair ou impair (TP)
  - Autres modes : Loops, Idle, Stop/Wait, half-duplex (Single-Wire)

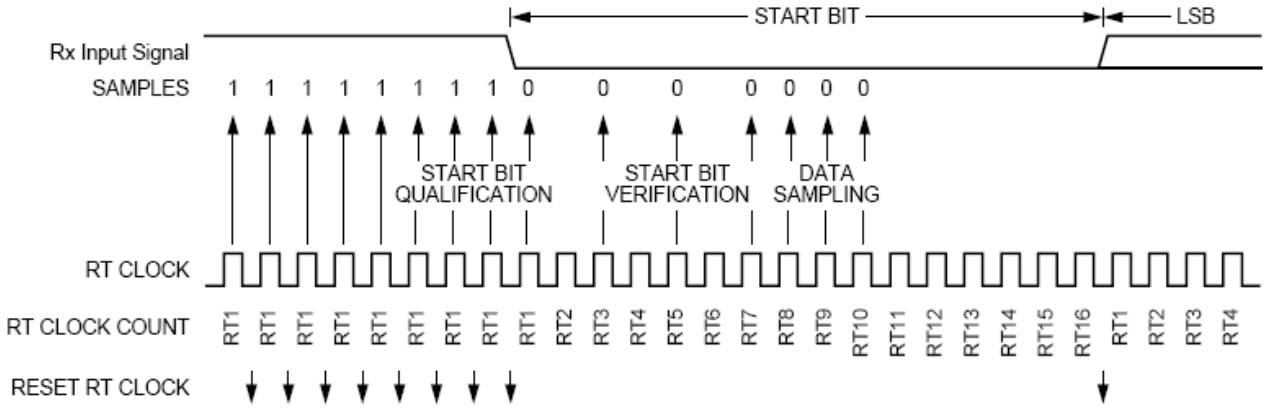
7.2 Module SCI

- **SCIx Control Register 2** :
  - Validation : Transmission (TE), Réception (RE)
  - Masques d'interruption :
    Registre d'émission vide (TIE), Émission terminée (TCIE),
    Registre de réception plein (RIE), Détection caractère Idle (ILIE)
  - Mode mise en veille (RWU)
  - Envoi caractère Break (SBK)

- **SCIx Status Register 1** :
  - Indicateurs d'interruption
    (mis à 0 par lecture de SCIxSR1 et lecture/écriture de SCIxDRL)
    Registre d'émission vide (TDRE), Émission terminée (TC),
    Registre de réception plein (RDRF), Détection caractère Idle (IDLE)
  - Détection d'erreurs :
    Parité (PF), Pas détection bit Stop (FE),
    Données bruitées (NF), Non lecture (OR)

7.2 Module SCI

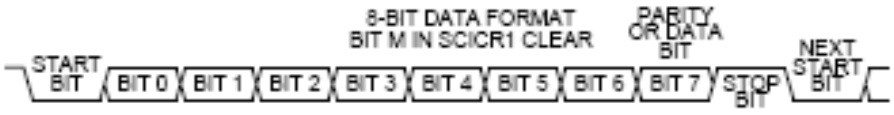
- **SCIx Status Register 1** :
  - Détection de bruit en réception (NF)
    - échantillonnage de Rx à une fréquence 16 fSCI
    - décision à partir des échantillons RT8 RT9 RT10



7.2 Module SCI

- **SCIx Status Register 2** :
  - Début de réception détecté (RAF)
  - Longueur caractère BRK (BRK13)
  - Direction TxD en mode single-wire (TXDIR)

- **SCIx Data Registers :**
  - Valeurs des données à envoyer (8 ou 9 bits)
    - Dans le cas d'un format 8 bits avec parité, la donnée est dans SCIxDRL avec R7 pour bit de parité



    - Dans le cas d'un format 9 bits, le bit R8 est dans SCIxDRH
  - Lecture des données (8 ou 9 bits)

7.3 Exemples

- **Programme d'émission-réception de chaîne ASCII sur SCI1**

```
; programme
        INCLUDE 'mc9s12dp256.inc'
        ORG $1300
DATA FCB 'HCS12',$0D,$0A
EOT             FCB $04; end of text
RDATA RMB 8
        ORG $1000
Entry: LDS #$3FFF; set up stack pointer
    BSR SCI1_INIT
LOOP: BSR SCI1_TRANSMIT
    BSR SCI1_RECEIVE
    CPX #EOT
    BNE LOOP
    BGND
```


7.3 Exemples

- **Programme d'émission-réception de chaîne ASCII sur SCI1**

```
SCI1_INIT:  MOVB #$08,DDRS; PS3/TxD1 as output  PS2/RxD1 as input
    MOVB #$0D,SCI1BDL; baud rate to 9600 if Bus clock = 2 MHz
    MOVB #$00,SCI1BDH
    MOVB #$02,SCI1CR1; eight-bit data, disable LOOP, parity, parity odd, Idle after start
    MOVB #$0C,SCI1CR2; no interrupt, transmitter enable, receiver enable
    LDAA SCI1SR1; 1st step to clear TDRE and RDRF
    STAA SCI1DRL; 2nd step to clear TDRE and RDRF
    LDX #DATA
    LDY #RDATA
    RTS
SCI1_TRANSMIT: LDAA 1,X+
    STAA SCI1DRL; load data register
WAIT_TDRE: BRCLR SCI1SR1,$80,WAIT_TDRE; wait for TDRE flag to 1
    STAA SCI1DRL; clear TDRE flag
    RTS
SCI1_RECEIVE: BRCLR SCI1SR1,$20,SCI1_RECEIVE; wait for RDRF flag to 1
    LDAA SCI1DRL
    ANDA #$7F  ; mask parity
    STAA 1,Y+; store data register
    RTS
```

7.3 Exemples

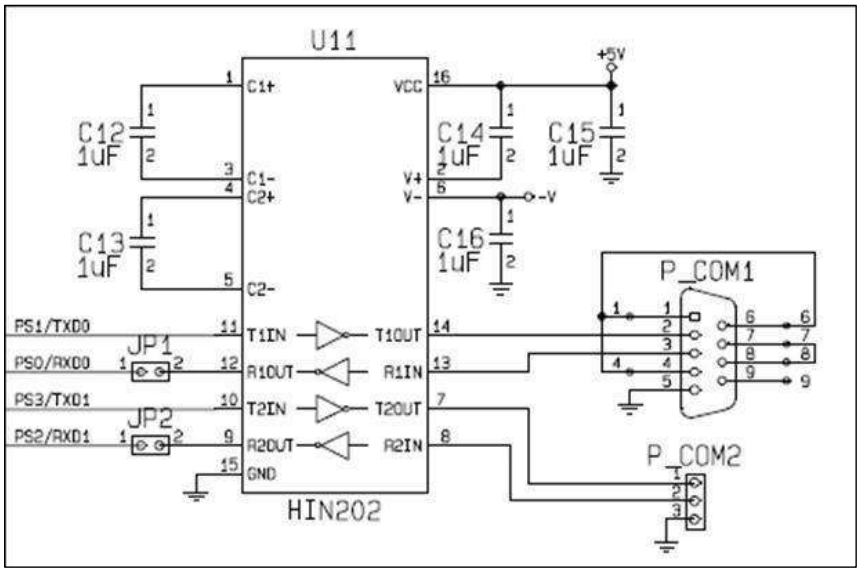
- Utilisation de la liaison série COM1 du PC vers SCI0
  - Sous-programmes *com1_routines.inc* :
    ONSCI : initialisation du module SCI0
    (9600 Baud, 8 bits, sans parité)
    OUTSTRG : envoi à l'écran du PC d'une chaîne de caractères
    pointée par X et terminée par le caractère EOT ($04)
    OUTCRLF : retour à la ligne du curseur de l'écran du PC
    OUTPUT : envoi d'un caractère contenu dans A
    INPUT : saisie dans A d'un caractère au clavier

  - Programme de test *main_com1_test.asm* :
    envoi de la chaîne de caractère *Do you enjoy HCS12 ? (O/N)*
    et saisie de la réponse

  - Utilisation du logiciel *Hyperterminal* du PC pour initier la communication avec COM1
    (Démarrer-Programmes-Accessoires-Communications)

7.3 Exemples

- Utilisation de la liaison série COM1 du PC vers SCI0
  - Adaptation RS232
    - Niveau 0 V en entrée ⟶ - 10 V en sortie
    - Niveau 5 V en entrée ⟶ 10 V en sortie



7.3 Exemples

```
;*****************************************
;   ROUTINES for COM1 / SCI0 communications *
; ONSCI                                    *
; OUTSTRG                                  *
; OUTCRLF                                  *
; OUTPUT                                   *
; INPUT                                    *
;*****************************************
;**************
;   EQUATES   *
;**************
EOT:  EQU  $04              ; end of text/table character
;***********************
; Initialize the SCI0 for 9600 baud
; Since we're using a 16.0 MHz clock and the Baud rate
      register is calculated:
;       BR = MCLK / (16 * Baud_Rate)
;       MCLK = crystal / 2
;       BR = 8,000,000 / (16 * 9600) which is 153,600
;       BR = 52 which is 34 hex
```

```
ONSCI:
      LDAA  #$0D          ; set baud rate to 9600 baud
      STAA  SCI0BDL       ; store low byte
      CLR   SCI0BDH       ; clear high byte
      LDAA  #$00 ; configure SCI0 control registers :
      eight-bit data, disable LOOP and parity
      STAA  SCI0CR1
       LDAA #$0C; enable transmit and receive
      staa  SCI0CR2
       LDAA                SCI0SR1  ; to clear TDRE
      flag (1)
      STD   SCI0DRH       ;                    (2)
      RTS
```


7.3 Exemples

```
;***********************
; Output string of ASCII bytes starting at x until end of text
      ($04).
OUTSTRG:
      JSR  OUTCRLF      ; output carriage-return
OUTSTRG0:
      PSHA          ; save a
OUTSTRG1:
      LDAA 0,X          ; read char into a
  CMPA #EOT     ; is this end of text?
  BEQ  OUTSTRG3       ; jump if yes
  JSR  OUTPUT         ; output character
  INX           ; incriment pointer
  BRA  OUTSTRG1  ; loop
OUTSTRG3:
      PULA              ; restore a
  RTS

;***********************
```

```
; Output a Carriage return
      and a line feed.  Returns a = cr.
OUTCRLF:
  LDAA #$0A       ; get LF
      JSR  OUTPUT       ; send it
      LDAA #$0D         ; get CR
      JSR  OUTPUT       ; send it
  LDAA #$00
      JSR  OUTPUT       ; output padding
      LDAA #$0D
      RTS
```

7.3 Exemples

```
;**********************
; Output A to SCI0
OUTPUT:
OUTSCI2:
      LDAB  SCI0SR1     ; read status
      BITB    #$80          ; test Transmit Data Register
      Empty bit
      BEQ     OUTSCI2    ; loop if TDRE=1
      ANDA #$7F          ; mask parity
      STAA   SCI0DRL    ; send character
 RTS
```

```
;**********************
; Input A from SCI0
INPUT:
INSCI2:
      LDAB  SCI0SR1     ; read status
      BITB    #$20 ; test Receive Data Register Empty bit
      BEQ     INSCI2       ; loop if RDRF=1
      LDAA  SCI0DRL     ; send character
 RTS
```


7.3 Exemples

```
 INCLUDE 'mc9s12dp256.inc' ; include register equates
;**************
;    DATA     *
;**************
      ORG     DATASTRT
BUF:  RMB 2
;**************
; TEXT TABLES *
;**************
MSG     FCC   'Do you enjoy HCS12 ? (O/N) '
      FCB   EOT

;**********************
```

```
      ORG     PROGSTRT
INCLUDE 'com1_routines.inc' ; include routines
Entry:
; If programming to Flash, uncomment the
      following line
      LDS     #$3FFF

      JSR ONSCI   ; initialize serial port
      LDX     #MSG          ; get message string
      JSR     OUTSTRG  ; send it out serial port
      JSR     OUTCRLF  ; output carriage-return
      JSR     INPUT         ; receive from serial port
      LDX #BUF
      STAA 0,X
      MOVB #EOT,1,X
;     JSR     OUTSTRG  ; send it out serial port
      JSR     OUTPUT; send char from serial port
;     JSR     OUTCRLF  ; output carriage-return
ENDPROG:
      BRA     ENDPROG  ; endless loop
```

7.3 Exemples

- Fonctionnement de *main_com1_test.asm* sur deux fenêtres
  - Ouvrez Hyperterminal
  - Sous Hyperterminal, établissez une connexion
  - Sous CodeWarrior, exécutez le programme compilé
  - La fenêtre Hyperterminal affiche la chaîne de caractères MSG
  - Saisissez votre réponse au clavier

7.4 Tests

1. Expliquez le rôle des bits TDRE et RDRF du registre SCIxSR1.

2. Quelle est la fonction du bit PF du registre SCIxSR1 ?

3. Quelles sont les interruptions associées au module SCIx ?

4. Modifiez le programme de l'exemple pour une vitesse de transmission de 14400 Baud.

5. Modifiez le programme pour un fonctionnement en interruption.

6. Décrivez le fonctionnement des transmissions full-duplex et half-duplex.

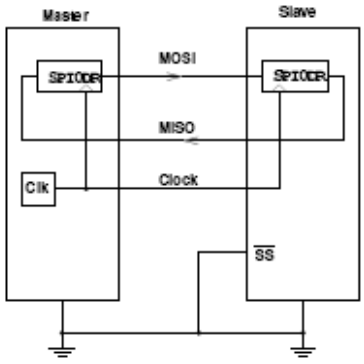
7. Qu'est ce qui différencie le module SPI du module SCI ?

Chapitre 8 : Interface série synchrone

- Sommaire

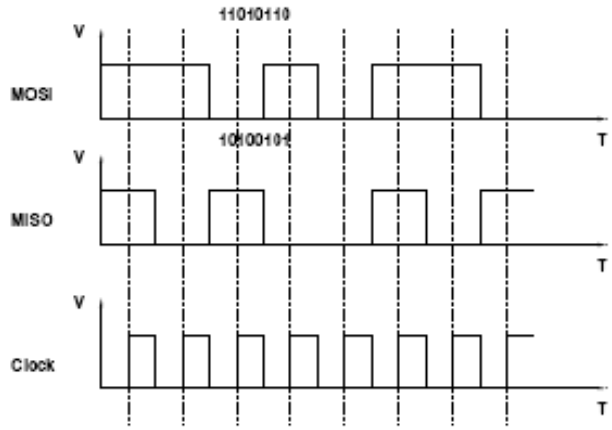
8.1 Interface série synchrone

- Caractéristiques
  - Ligne d'horloge générée par le maître
  - Affectation d'un périphérique en esclave ($\overline{\text{SlaveSelect}} = 0$)
  - Registre à décalage 8 bit de chaque côté de la ligne (Data Register)
  - Ligne de transmission de l'émetteur maître vers esclave récepteur (MOSI Master Output Slave Input)
  - Ligne de transmission de l'émetteur esclave vers maître récepteur (MISO Master Input Slave Output)



8.1 Interface série synchrone

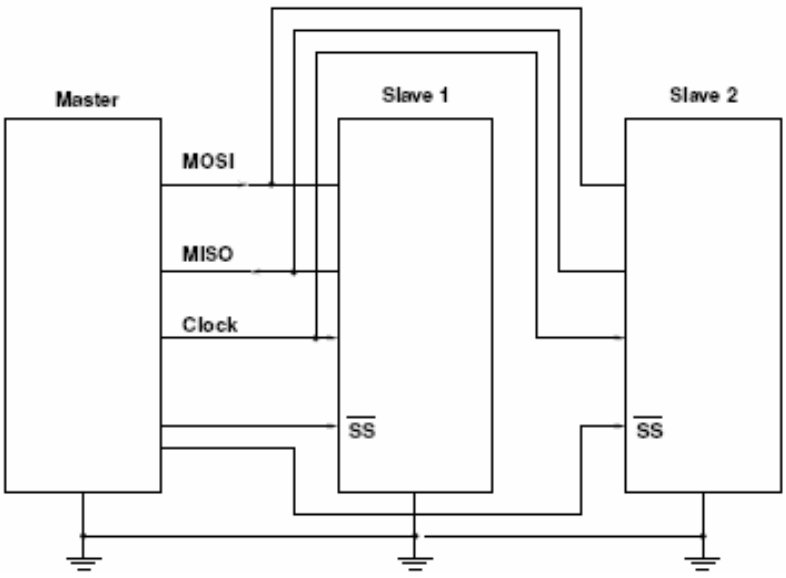
- Fonctionnement
  - Exemple de transmission d'une donnée %11010110 sur MOSI avec front actif montant de l'horloge



  - Envoi des bits successifs MSB à LSB (ou LSB à MSB) en 8 périodes
  - Largeur d'un bit : inverse de la vitesse de transmission (Baud ou bit/s)

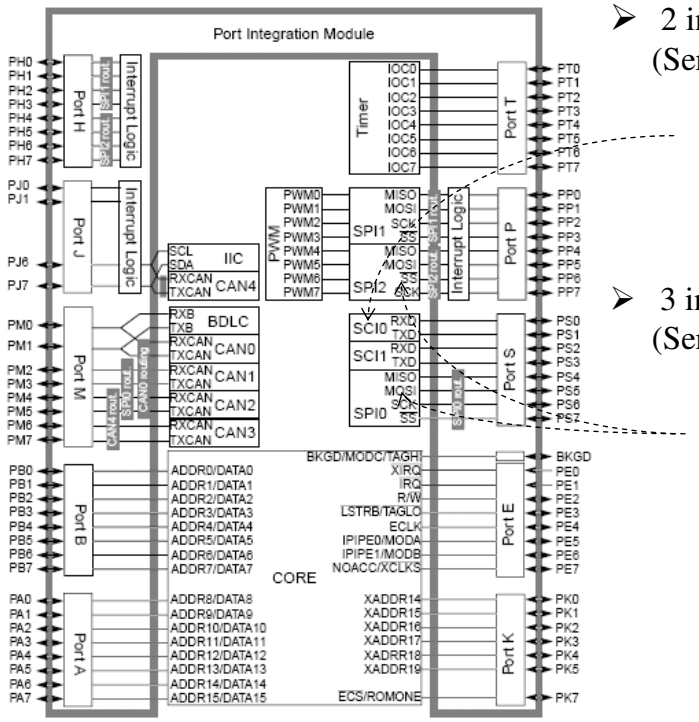
8.1 Interface série synchrone

- Système de communication : esclaves multiples, maître unique
  - Communication full-duplex (émission et réception simultanées)



8.1 Interface série synchrone

- 5 interfaces séries indépendantes pour le HCS12 :



➢ 2 interfaces série asynchrones (Serial Communications) :

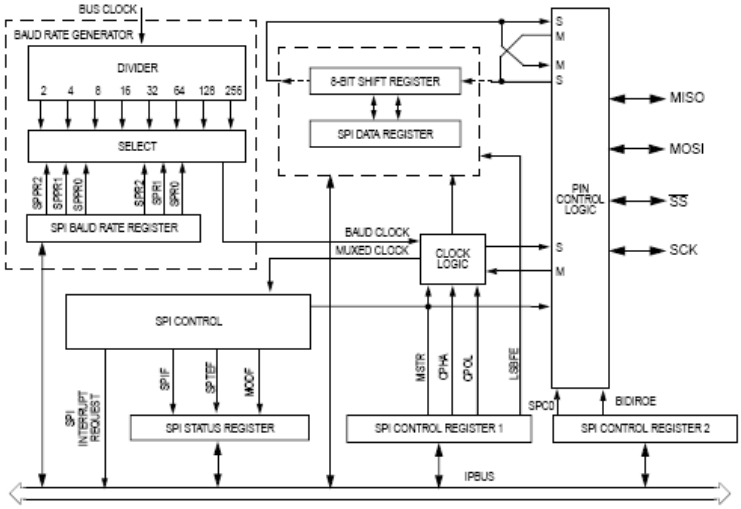
- SCI0, SCI1 (port S)
- compatibles RS232

➢ 3 interfaces série synchrones (Serial Peripheral) :

- SPI0 (port S), SPI1et SPI2 (port P)
- horloge
- communications avec périphériques et microcontrôleurs

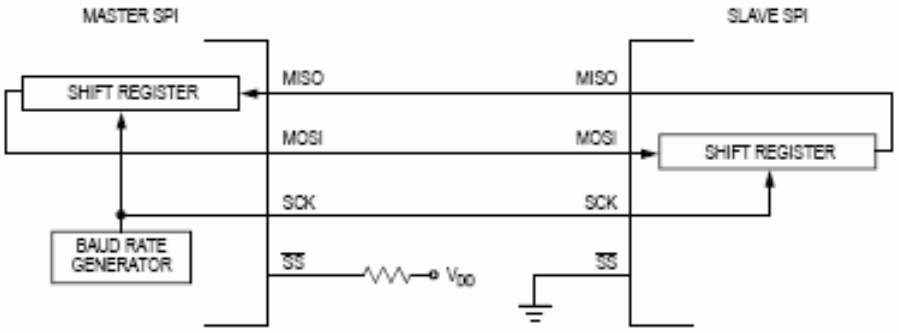
8.2 Module SPI

- Architecture de SPI0 (Serial Peripheral Interface)
  - ➢ Générateur d'horloge SCK (PS6) sur la base de Bus Clock
  - ➢ Lignes de transmission MISO (PS4) et MOSI (PS5)
  - ➢ Sélection de l'esclave $\overline{SS}$ (PS7)
  - ➢ Registre à décalage 8 bits



8.2 Module SPI

- Câblage du système émetteur / récepteur
  - Choix matériel maître ou esclave pour le HCS12 et pour le périphérique
  - Programmation de l'option maître / esclave



8.2 Module SPI

- Registres pour SPI0 ($D8-$DF)
  - Registre de vitesse de transmission (SPI0BR)
  - Registres de contrôle (SPI0CR1 et SPI0CR2)
  - Registre d'état (SPI0SR)
  - Registre 8 bits de données (SPI0DR)

Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPI0CR1	Read: Write:	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
SPI0CR2	Read:	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
	Write:								
SPI0BR	Read:	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
	Write:								
SPI0SR	Read:	SPIF	0	SPTEF	MODF	0	0	0	0
	Write:								
Reserved	Read:	0	0	0	0	0	0	0	0
	Write:								
SPI0DR	Read: Write:	Bit7	6	5	4	3	2	1	Bit0

8.2 Module SPI

- **SPIx Baud rate Register** :
  - Fréquence d'horloge

    Fsck = Bus CLK / ((SPPR+1) 2^{SPR+1})

    avec SPR sur 3 bits et SPPR sur 3 bits

ex.: SPR = 2 ; SPPR = 0 ; Bus CLK = 2 MHz ; Fsck = 500 kHz

- **SPIx Control Register 1** :
  - Validation SPIx (SPE)
  - Masque d'interruption (SPIE), interruption à l'émission (SPTIE)
  - Mode : Maître/Esclave (MSTR), validation de sortie SS (SSOE)
  - Horloge : polarité (CPOL) et retard (CPHA)
  - 1er bit à envoyer (LSBFE)

8.2 Module SPI

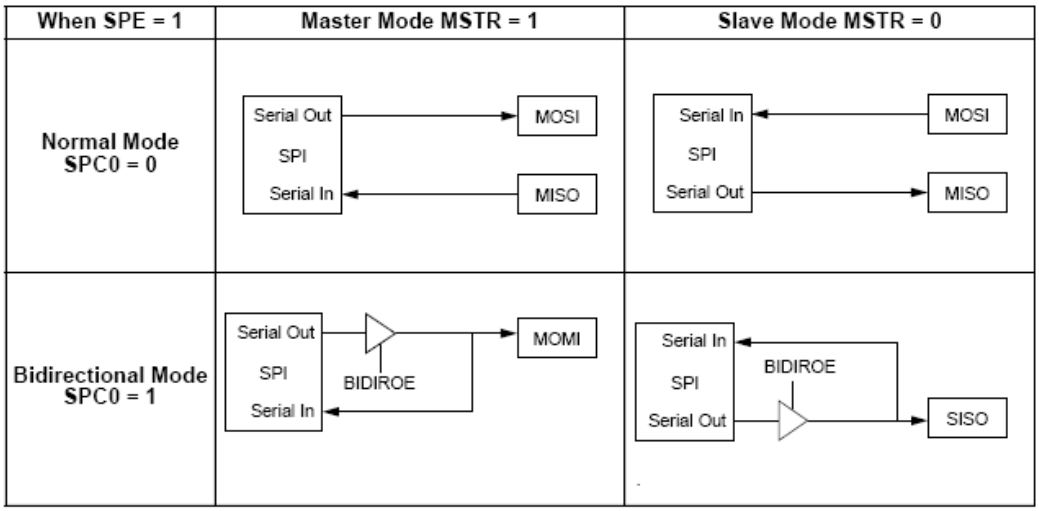
- **SPIx Control Register 2** :
  - Détection d'erreur de fonctionnement : On / Off (MODFEN)
  - Validation du buffer de sortie en mode bidirectionnel (BIDIROE)
  - Arrêt de l'horloge en mode attente (SPISWAI)
  - Mode de transmission bidirectionnelle (SPC0)

Pin Mode		SPC0	MSTR	MISO [1]	MOSI [2]	SCK [3]	$\overline{SS}$ [4]
A	Normal	0	0	Slave Out	Slave In	SCK in	$\overline{SS}$ in
B			1	Master In	Master Out	SCK out	$\overline{SS}$ I/O
C	Bidirectional	1	0	Slave I/O	—	SCK in	$\overline{SS}$ In
D			1	—	Master I/O	SCK out	$\overline{SS}$ I/O

NOTES:
1. Slave output is enabled if BIDIROE bit = 1, $\overline{SS}$ = 0, and MSTR = 0 (C)
2. Master output is enabled if BIDIROE bit = 1 and MSTR = 1 (D)
3. SCK output is enabled if MSTR = 1 (B, D)
4. $\overline{SS}$ output is enabled if MODFEN bit = 1, SSOE = 1, and MSTR = 1 (B, D).

8.2 Module SPI

- Mode de transmission bidirectionnelle (MOMI ou SISO)



8.2 Module SPI

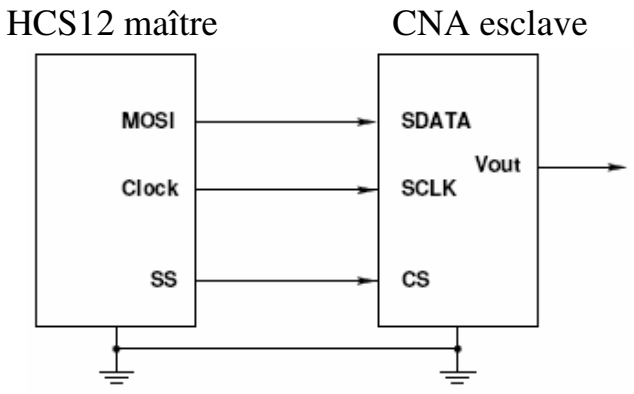
- **SPIx Status Register** :
  - Indicateur d'interruption (SPIF)
    (mise à 0 par lecture de SPIxSR et lecture ou écriture de SPIxDR)
  - Indicateur de registre d'émission vide (SPTEF)
    (mise à 0 par lecture de SPIxSR et écriture de SPIxDR)
  - Conflit de choix (MODF) entre maître (MSTR) et esclave (SS)

- **SPIx Data Register** :
  - Données 8 bits à émettre ou recevoir

- **Data Direction Register for port S (SPI0) ou P (SPI1, SPI2)**
  - Direction des données PS4 à PS7 (MISO, MOSI, SCK, /SS) :
    1 en sortie, 0 en entrée

8.3 Exemples

- **Sortie des données numériques vers convertisseur CNA**
  - Contrôle du périphérique (ChipSelect) par SS (port PS7)
  - Donnée série SDATA liée à MOSI (port PS5)
  - Horloge série SCLK liée à SCK (port PS4)



8.3 Exemples

- **Programme d'émission-réception de chaîne ASCII sur SPI0**

```
; programme
        INCLUDE 'mc9s12dp256.inc'
        ORG $1300
DATA FCB 'HCS12',$0D,$0A
EOT FCB $04; end of text
RDATA RMB 8
        ORG $1000
Entry: LDS #$3FFF; set up stack pointer
    BSR SPI0_INIT
LOOP: BSR SPI0_TRANSMIT
    BSR SPI0_RECEIVE
    CPX #EOT
    BNE LOOP
    BGND
```

8.3 Exemples

- **Programme d'émission-réception de chaîne ASCII sur SPI0**

```
SPI0_INIT: BSET PTS,$80; set line /SS to 1 to disable slave
      MOVB #$E0,DDRS; /SS / PS7, SCK / PS6 and MOSI / PS5 as outputs, MISO / PS4 as input
      MOVB #$02,SPI0BR; SCK to 500 kHz if Bus clock = 2 MHz
      MOVB #$52,SPI0CR1; no interrupt, HCS12 master, SS output, MSB first on first rising clock edge,
      MOVB #$10,SPI0CR2; normal transmission, MODFEN
      LDAA SPI0SR; 1st step to clear SPIF flag
      LDAA SPI0DR; 2nd step to clear SPIF
      LDX #DATA
      LDY #RDATA
      BSET SPI0CR1,$40; enable SPI SPE = 1
      RTS
;
SPI0_TRANSMIT: LDAA 1,X+
      BCLR PTS,$80; /SS = 0 to start
      STAA SPI0DR; load data register
WAIT: BRCLR SPI0SR,$80,WAIT; wait for SPIF
      BSET PTS,$80
      RTS
```

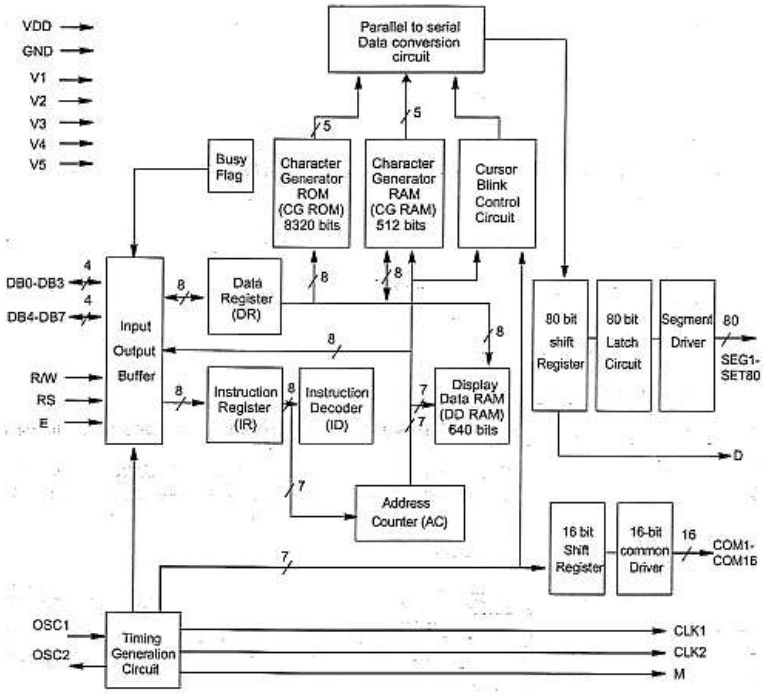

8.3 Exemples

- **Programme d'émission-réception de chaîne ASCII sur SPI0**

```
SPI0_RECEIVE: BRCLR SPI0SR,$80,SPI0_RECEIVE; wait SPIF flag
      LDAA SPI0DR
      STAA 1,Y+; store data register
      RTS
      END
```

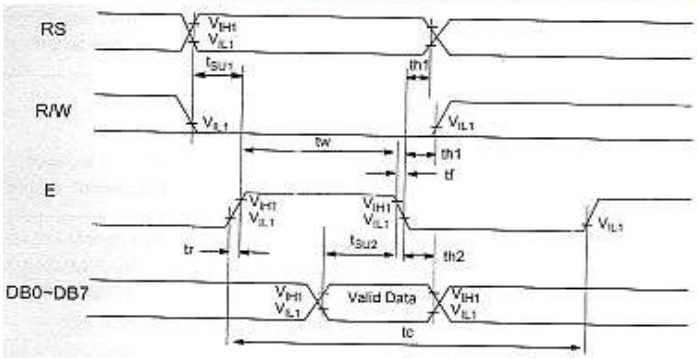
8.3 Exemples

- **Liaison avec afficheur Samsung KS0070B**

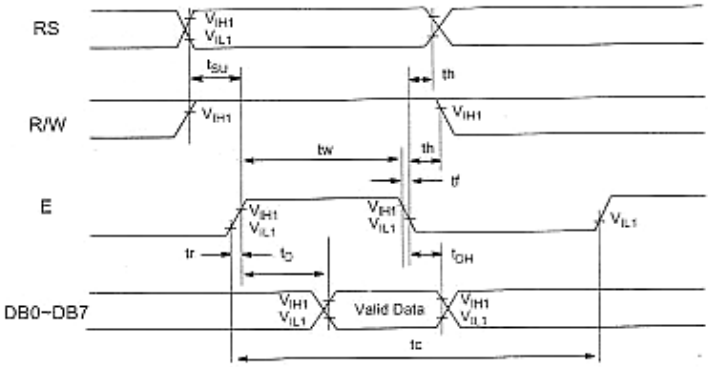


8.3 Exemples

- Écriture



- Lecture

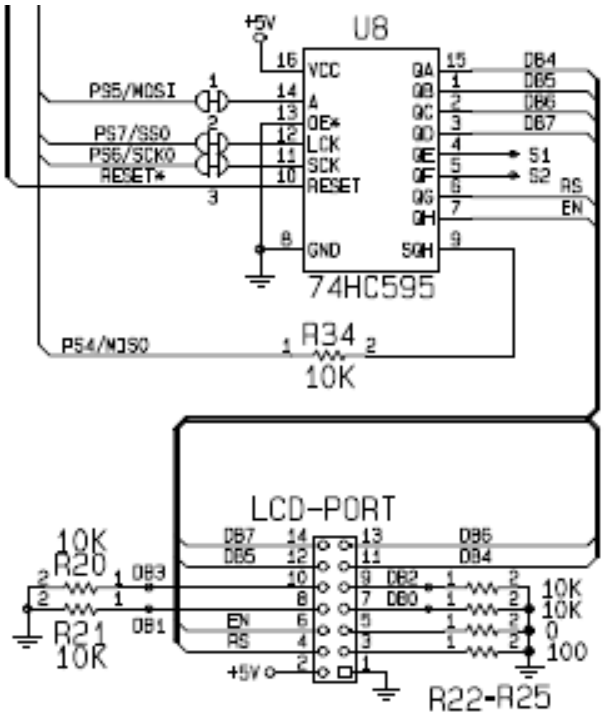


8.3 Exemples

- **Interface SPI vers port LCD 4 bits parallèle sur la carte**

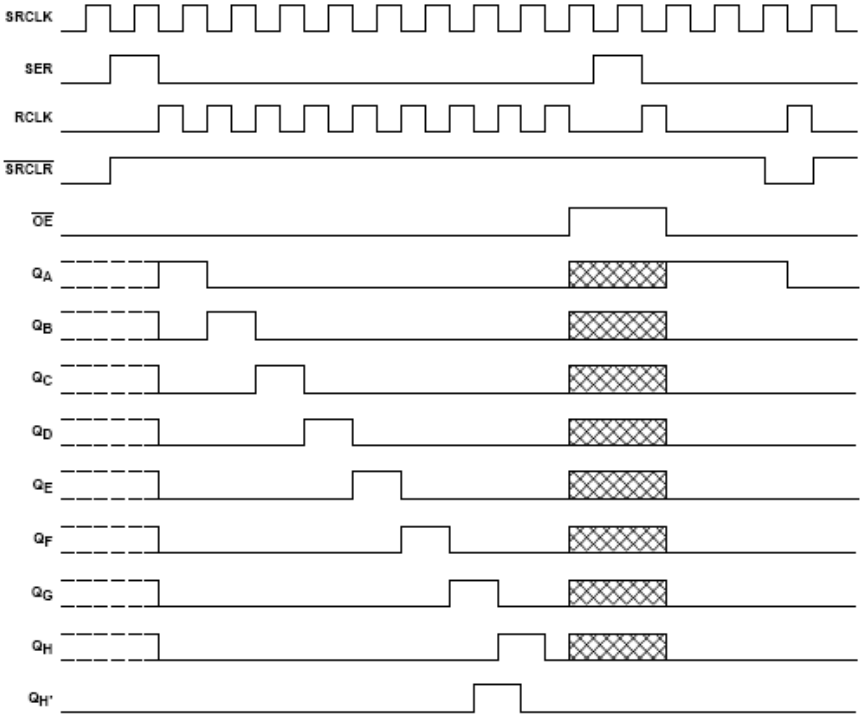
Registre à décalage
Mode écriture seule

3 Transferts :
- Transfert D0-D3 et RS (Register Select)
- Transfert avec EN à 1
- Transfert avec EN à 0



8.3 Exemples

- **Interface SPI vers port LCD 4 bits parallèle sur la carte**



8.3 Exemples

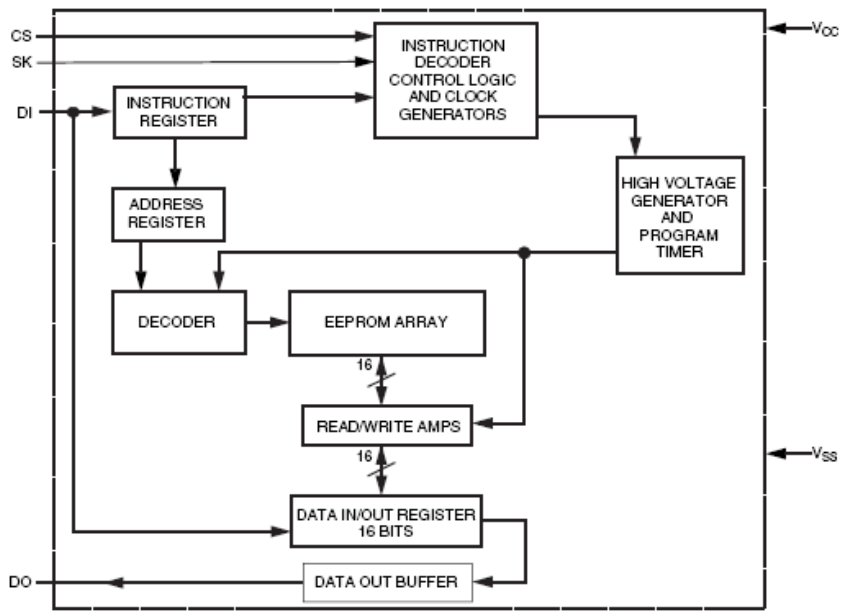
- **Interface SPI vers port LCD 4 bits parallèle sur la carte**

```
; Lcd Write 4 bit Data , lower 4 bits of acc A
;     0 = Lcd-Db4      4 =
;     1 = Lcd-Db5      5 =
;     2 = Lcd-Db6      6 = EN
;     3 = Lcd=Db7      7 = RS
LCD_WR_4:         pshx
        psha                    ; save value
        ldaa    LCDBUF   ; load Lcd buffer
        anda    #$f0          ; get only controls signals
        pulb                    ; get data
        andb    #$0f            ; only bits 0-3 are data
        aba
        staa    LCDBUF   ; save data & lcd signals
        ldx     #LCDBUF
        bclr    0,x,EN          ; enable low
        jsr     LCD_SEND        ; send the data
        bset    LCDBUF,EN       ; enable high
        jsr     LCD_SEND
        bclr    0,x,EN          ; enable low
        jsr     LCD_SEND
        pulx
        rts
```

```
; Lcd Send
LCD_SEND
        ldaa    LCDBUF          ; load data & signals
        jsr     SS_IO           ; send to lcd
        rts
;
;-------------------------------------------------
; Simple Serial Driver (SPI), source code...
; Acc A is send serially, msb first
; return received in Acc A
SS_IO:
      jsr  DELAY10M
      ldab     SPI0SR   ; CLEAR STATUS OF SPI
      staa     SPI0DR  ; SEND / RECEIVE BYTE
SS_IO_LP:     brclr SPI0SR,#$80,SS_IO_LP
; WAIT IF NO FLAG
      ldaa    SPI0DR          ; read receive value
       rts
```

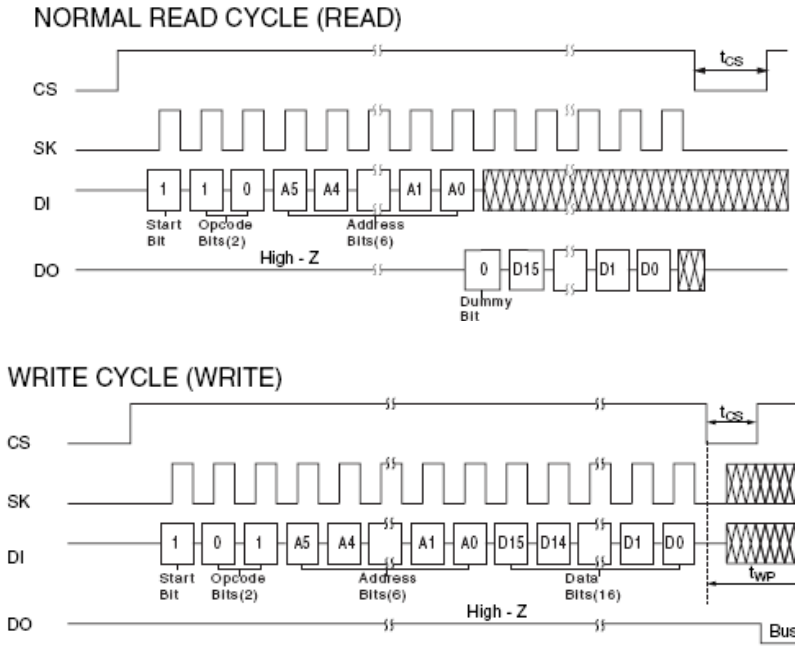

8.3 Exemples

- **Liaison avec puce EEPROM FM93C46**



8.3 Exemples

- **Liaison avec puce EEPROM FM93C46**



8.4 Tests

1. Expliquez le rôle des bits SPIF et SPTEF du registre SPIxSR.

2. Quelle est la fonction du bit MODF du registre SPIxSR ?

3. Quelles sont les interruptions associées au module SPIx ?

4. Modifiez le programme de l'exemple pour une horloge à 125 kHz.

5. Modifiez le programme pour un fonctionnement en interruption.

6. Décrivez le fonctionnement des transmissions bidirectionnelles.

7. Quelle procédure doit-on utiliser pour mettre à zéro les drapeaux du registre d'état SPIxSR ?