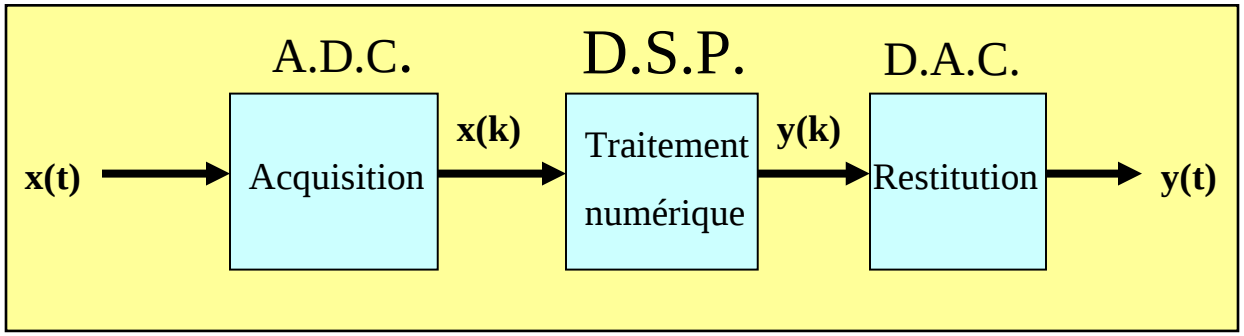
D.S.P.
(Digital Signal Processor)

- Processeur destiné au traitement du signal

- Les grandeurs physiques sont transformées en nombres

- Architecture des DSP optimisée pour le traitement numérique du signal



PROGRESSION

- Principaux traitements et domaines d'utilisation
- Comment choisir un D.S.P.
- Chaîne d'acquisition traitement restitution
- Les convertisseurs analogiques numériques
- Les convertisseurs numériques analogiques
- Etude du DSPIC 30F6014
- Calcul, Simulation et Programmation de filtre R.I.F. et R.I.I.
- Application au traitement du son.

PRINCIPAUX TRAITEMENTS

- Analyser un signal (FFT)

- Détecter, identifier des signaux (1750, DTMF, parole, image…)

- Filtrer

- Générer des signaux

- Moduler démoduler (modem, ADSL, Ethernet, AX25….)

- Crypter, décrypter

DOMAINES D ’APPLICATION

- Télécommunications

- Sons, Images, Vidéo

- Instrumentation

- Biomédical

- Commande de process

- Equipement automobile, avionique, radars...

COMPARAISON AUX AUTRES DISPOSITIFS DE TRAITEMENTS

Traitements microprogrammés

- microordinateurs (Traitements complexes mais rarement en temps réel)
- microprocesseurs classiques (<100 Kéch./s)
- DSP (< 1Méch/s)

Traitements câblés (architectures parallèles)

- FPGA,... <50 Méch/s...
- ASIC (réservé aux grosses productions)

COMMENT CHOISIR UN D.S.P.

Caractéristiques communes

- Architecture optimisée pour le traitement du signal (nombreux bus,Harvard, pipe-line…)
- Instruction multiplication accumulation en un cycle
- Modes d ’adressages particuliers

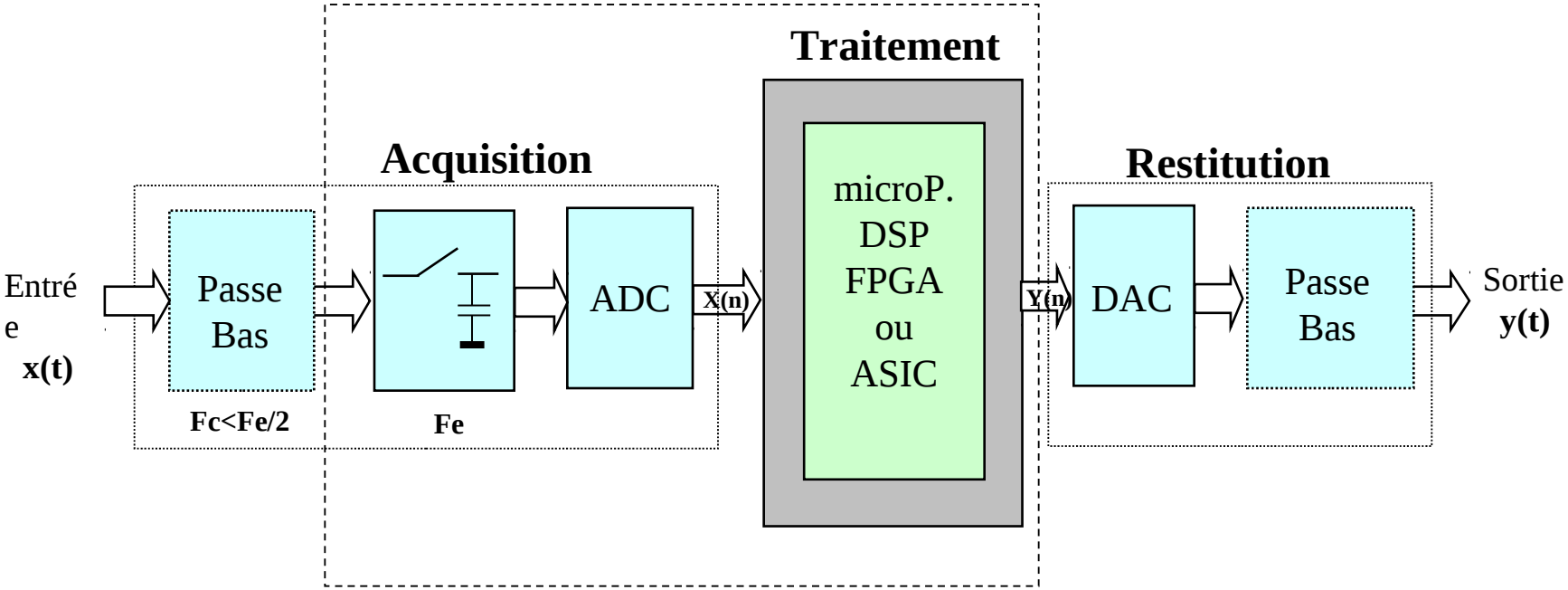
Les différentes familles de DSP

- DSP avec unité de calculs en virgule fixe
- DSP avec unité de calculs en virgule flottante câblé
- DSP avec des périphériques intégrés (contrôle moteur..)

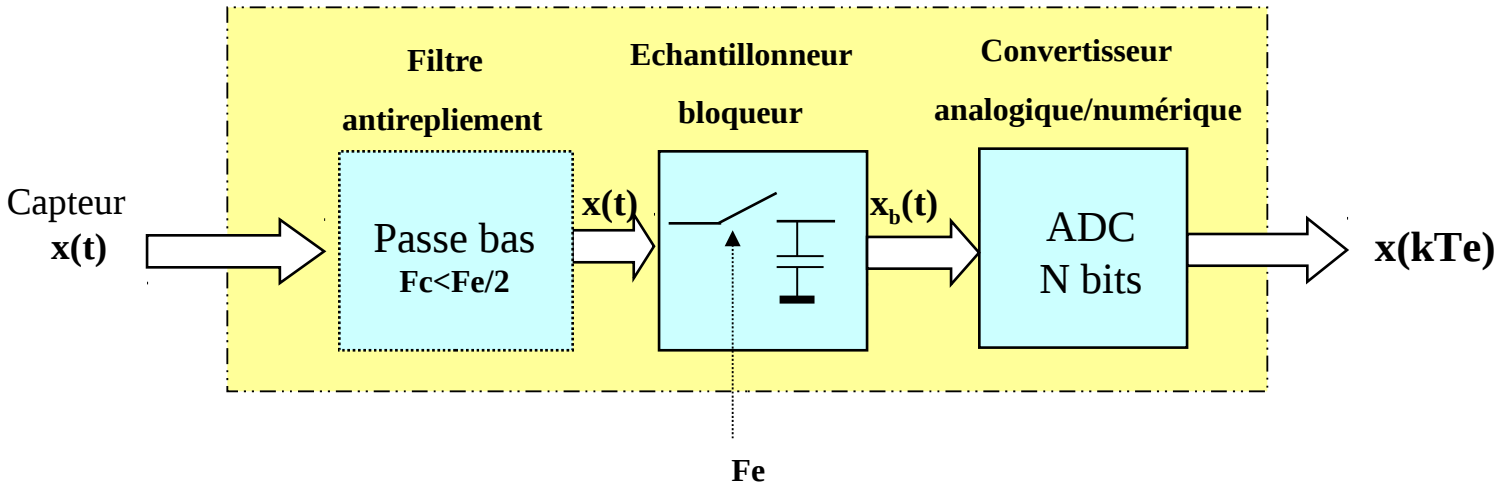
Les langages de programmation: L.machine, C, G. de C(ViSim..)

Constructeurs : AT&T, Texas Inst., Motorola, Analoge Devices...

ACQUISITION TRAITEMENT RESTITUTION



ACQUISITION



ACQUISITION

Produire une suite de nombre entiers x(k) à partir du signal analogique x(t)

Filtre antirepliement du spectre (anti-aliasing

- Choix de la fréquence d ’échantillonnage (Shannon)
- Choix de la fréquence de coupure du filtre passe-bas

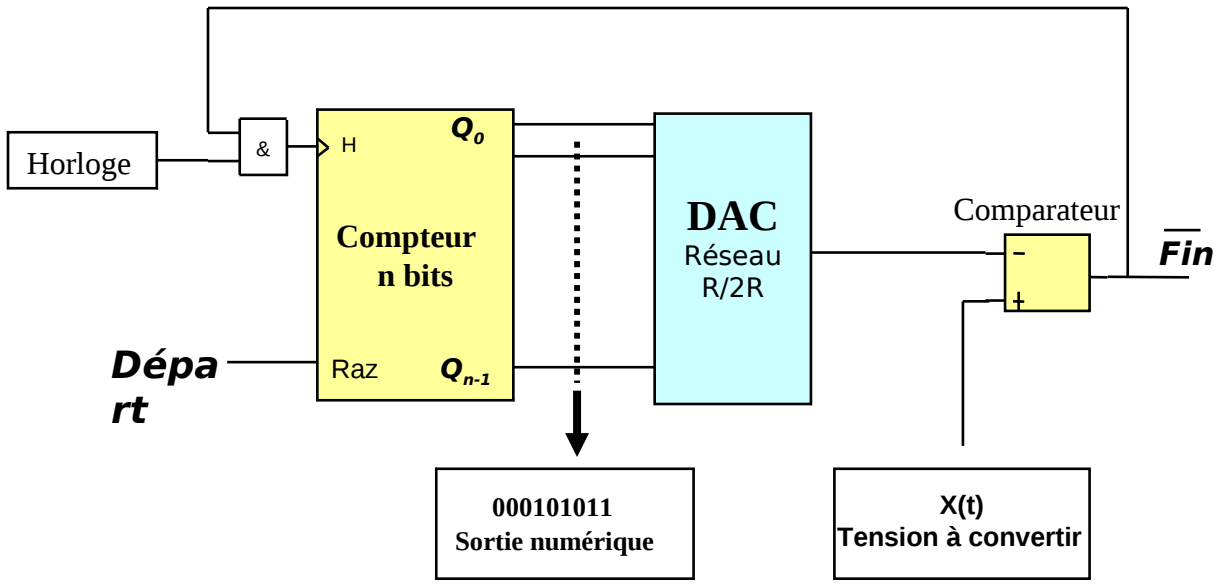
Echantillonneur-bloqueur

- Rôle
- Problèmes liés à l ’impédance

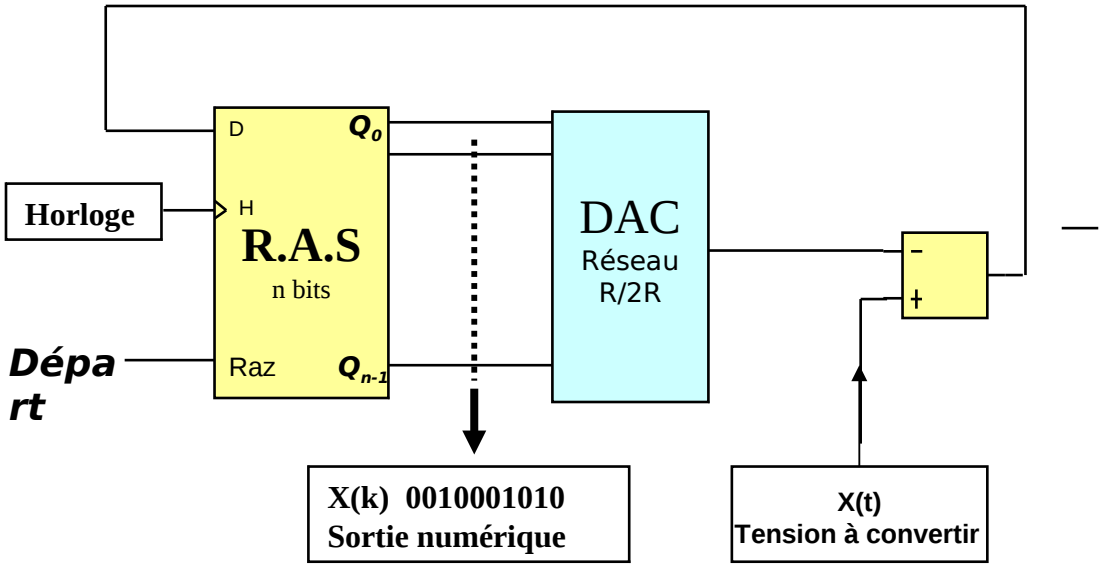
Quantification

- Résolution du convertisseur (A.D.C.)
- Temps de réponse
- Linéarité
- Précision et dynamique

Convertisseur Analogique - Numérique à rampe numérique



Convertisseur Analogique – à registre à approximations successives



Autres convertisseurs Analogique - Numérique

Convertisseur « Flash »

Convertisseur « sigma-delta »

(voir TD 1)

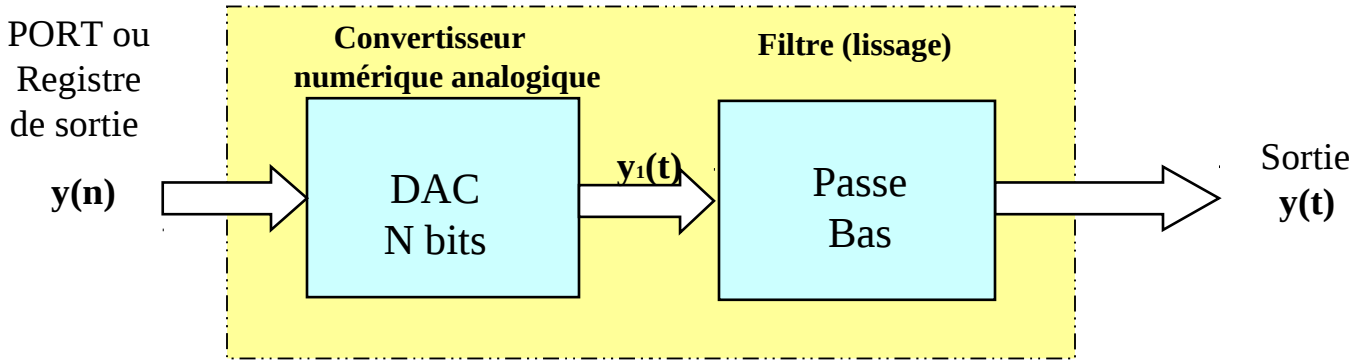
RESTITUTION

- Bloqueur d ’ordre 0 (registre de sortie)

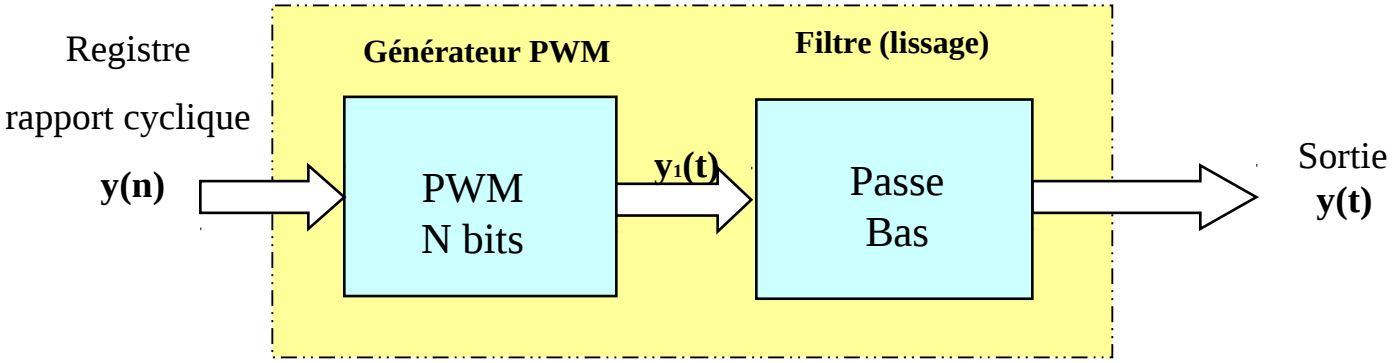
- DAC ou MLI (PWM)

- Filtre passe-bas analogique

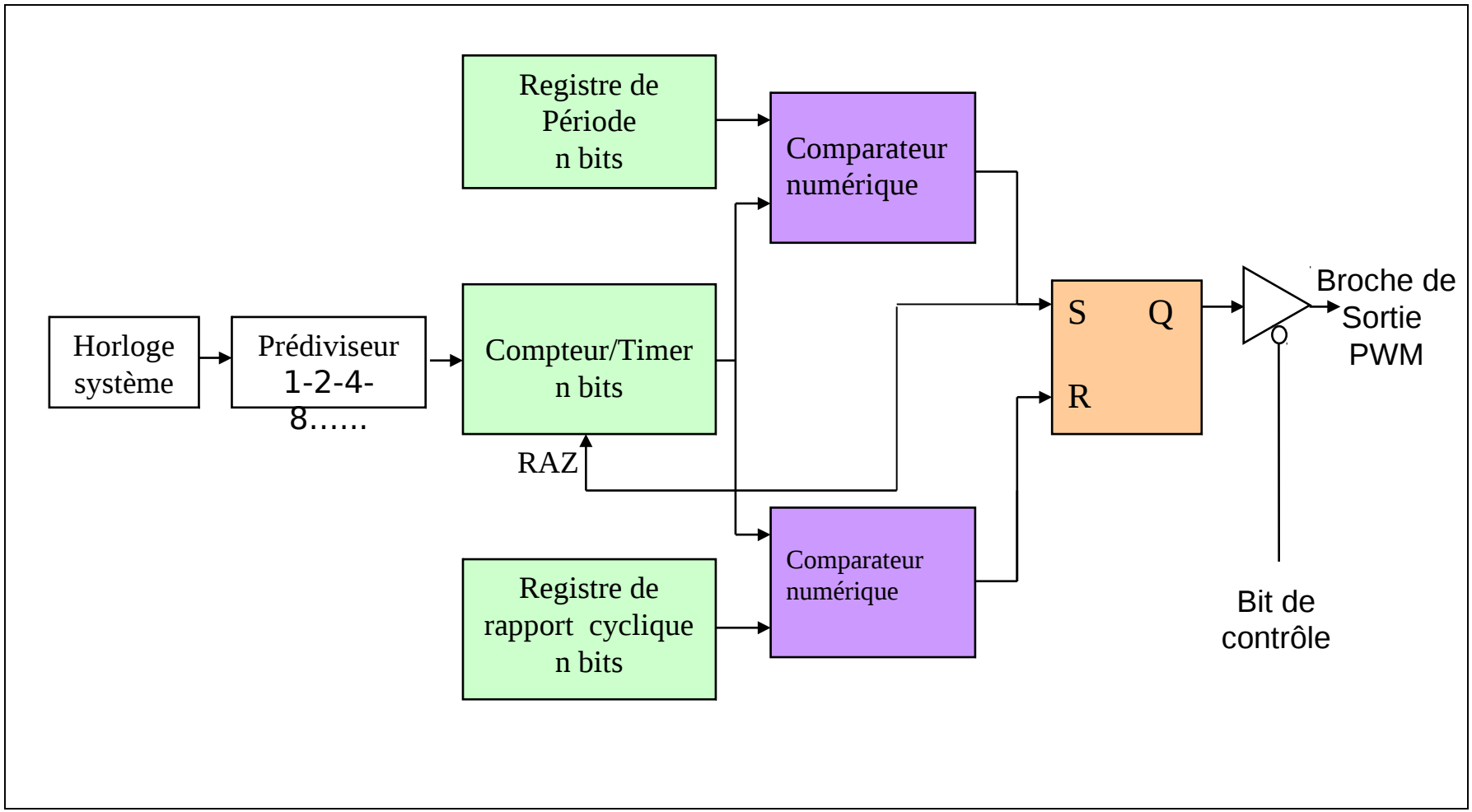
RESTITUTION
convertisseur analogique/numérique



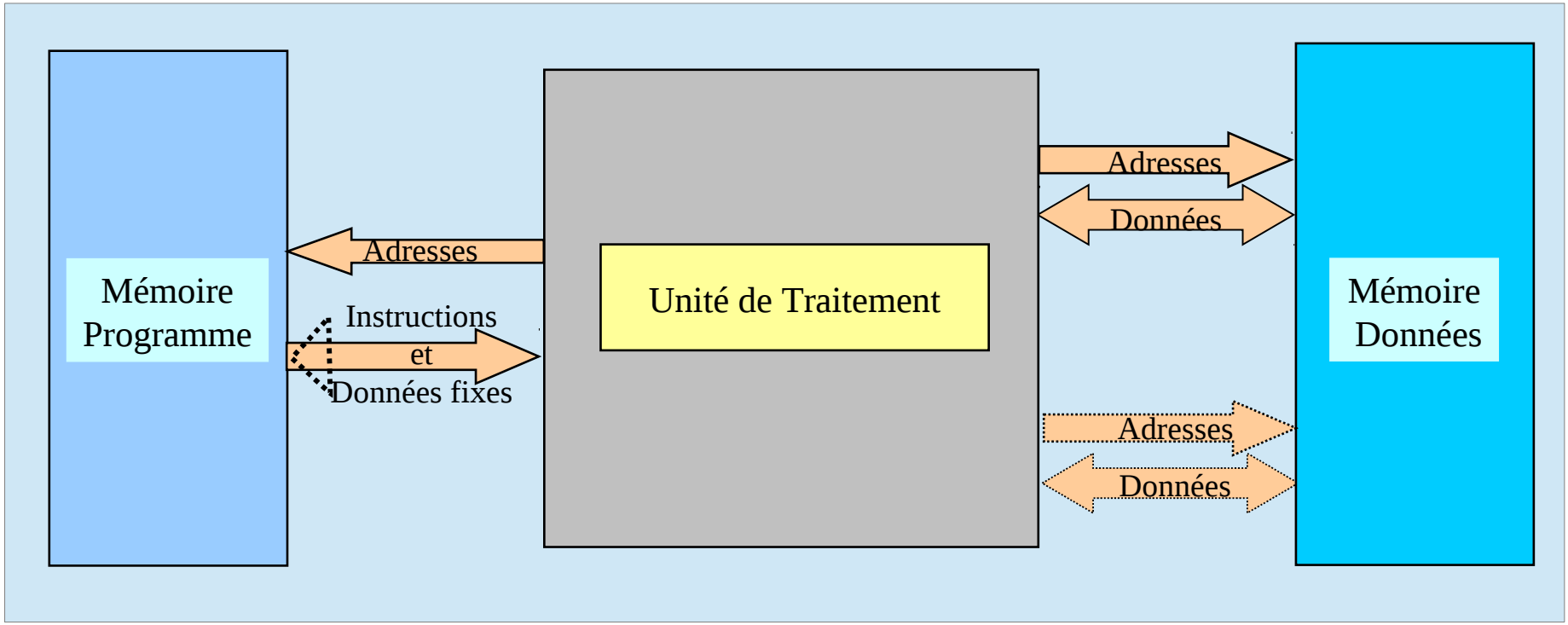
RESTITUTION
Par modulation de largeur d'impulsions



Génération PWM (Principe)



Architecture Harvard des DSPic (simplifiée)



PRESENTATION du DSPIC 33FJ16MCxxx (Microchip)

Documentation technique

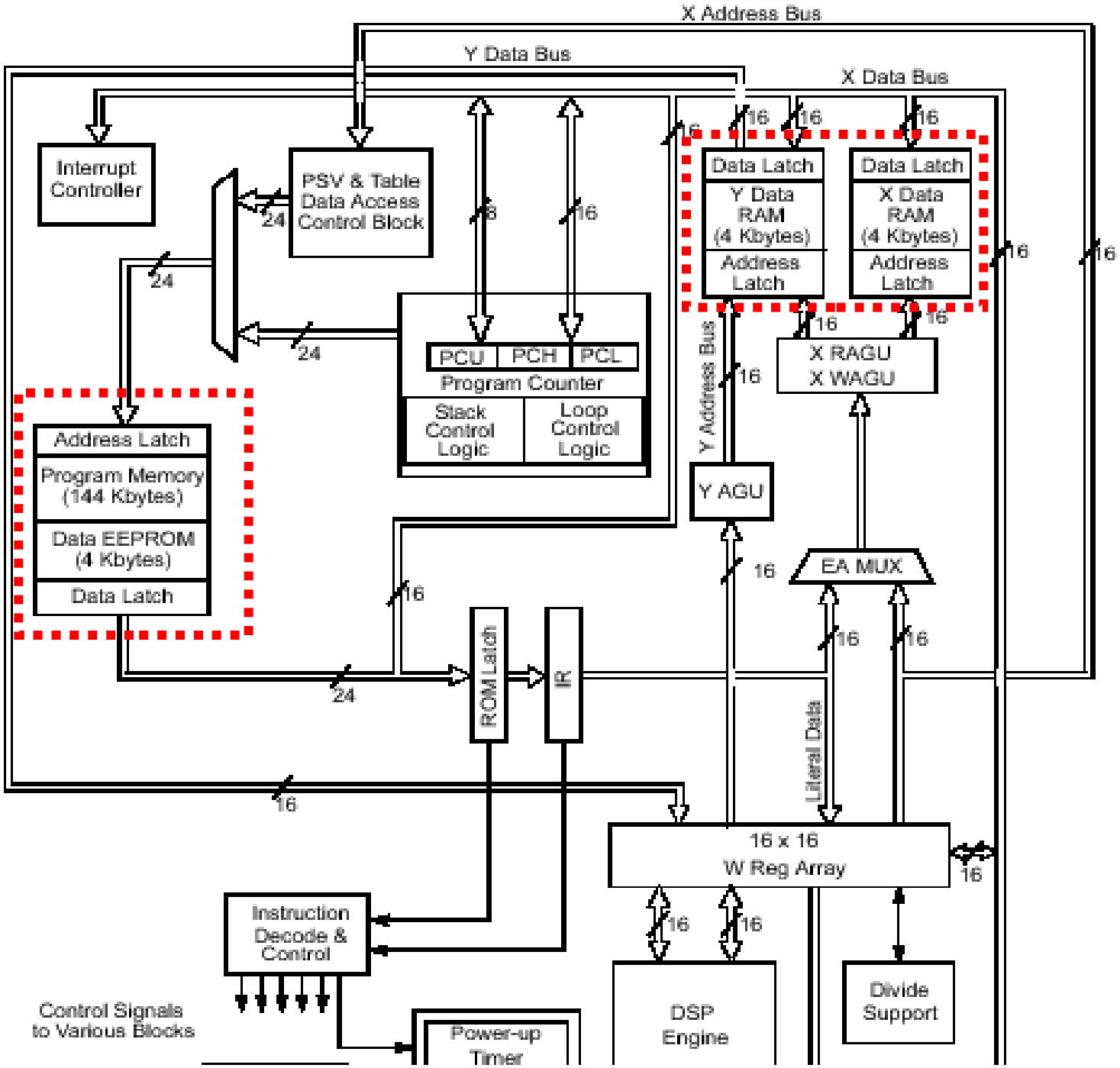
- Datasheet principal 33FJ16MC :	**70283k.pdf**
- CPU:	**70204c.pdf**
- Instructions :	**70157f.pdf**
- Ports (Entrées/Sorties)	**70193d.pdf**
- ADC :	**70183d.pdf**
- Timers	**70205d.pdf**
- uart	**70188e.pdf**
- Interruptions	**70214c.pdf**
- Horloge	**70186e.pdf**
- Watchdog	**70196d.pdf**
- Librairies	**51456g.pdf**
- Assembleur Linker :	**51317h.pdf**
- Compilateur XC16 :	**50002071C.pdf**

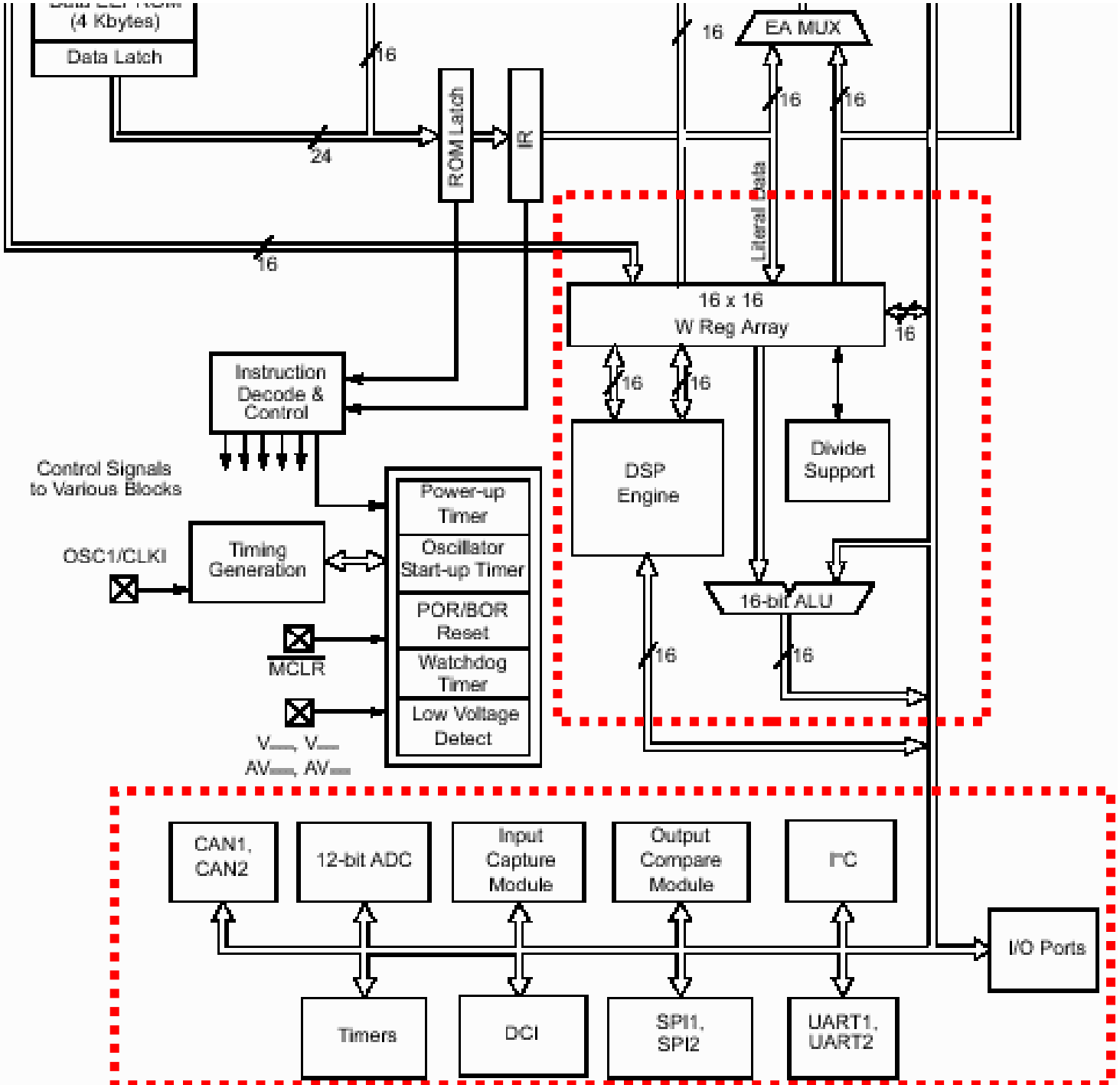
PRESENTATION du DSPIC 33f (Microchip)

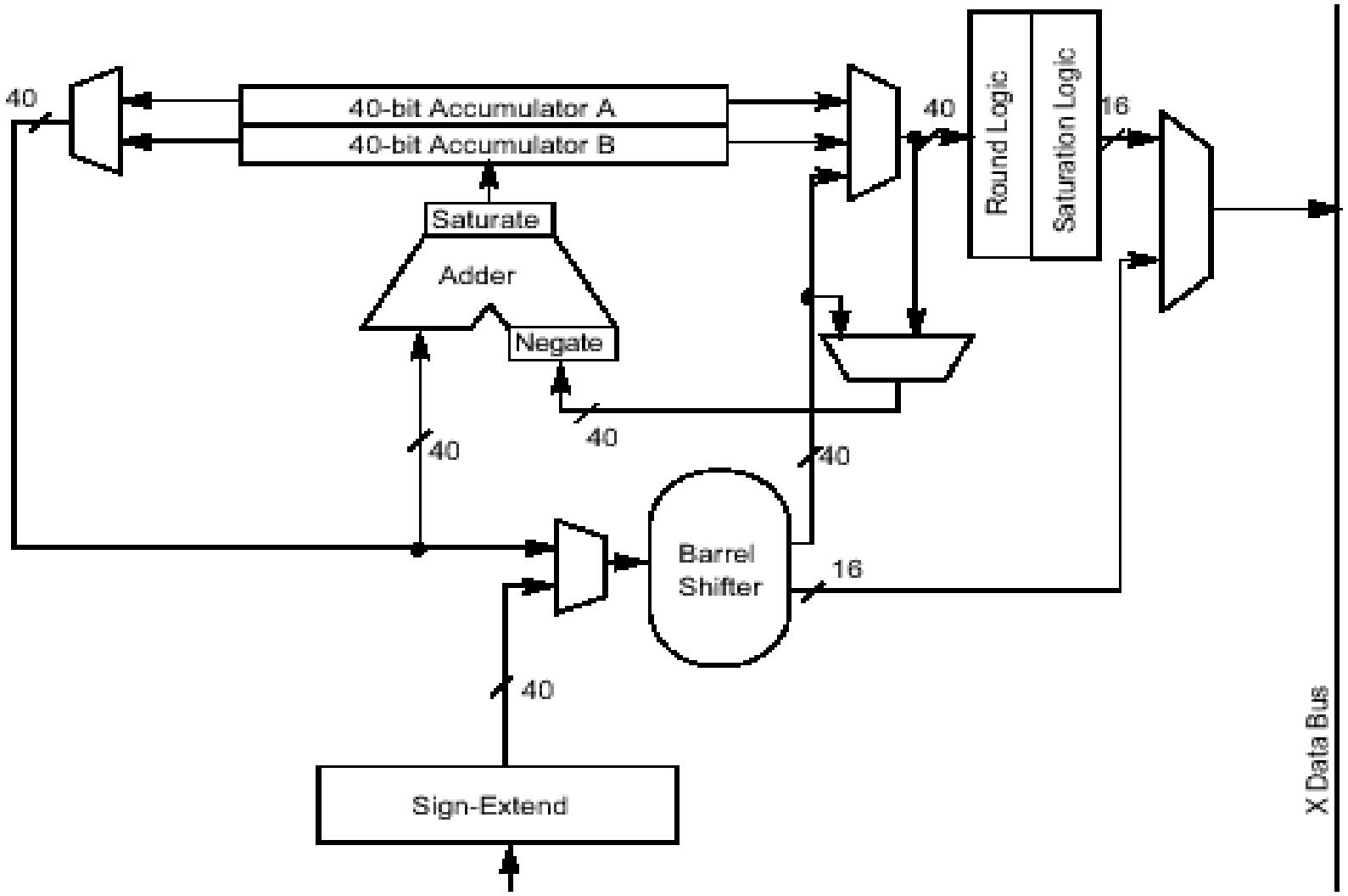
- Architecture
- Registres
- Mémoire
- Instructions et Modes d ’adressages
- Les principales fonctionnalités
- Les Timers
- Le convertisseur AD
- Gestion des interruptions

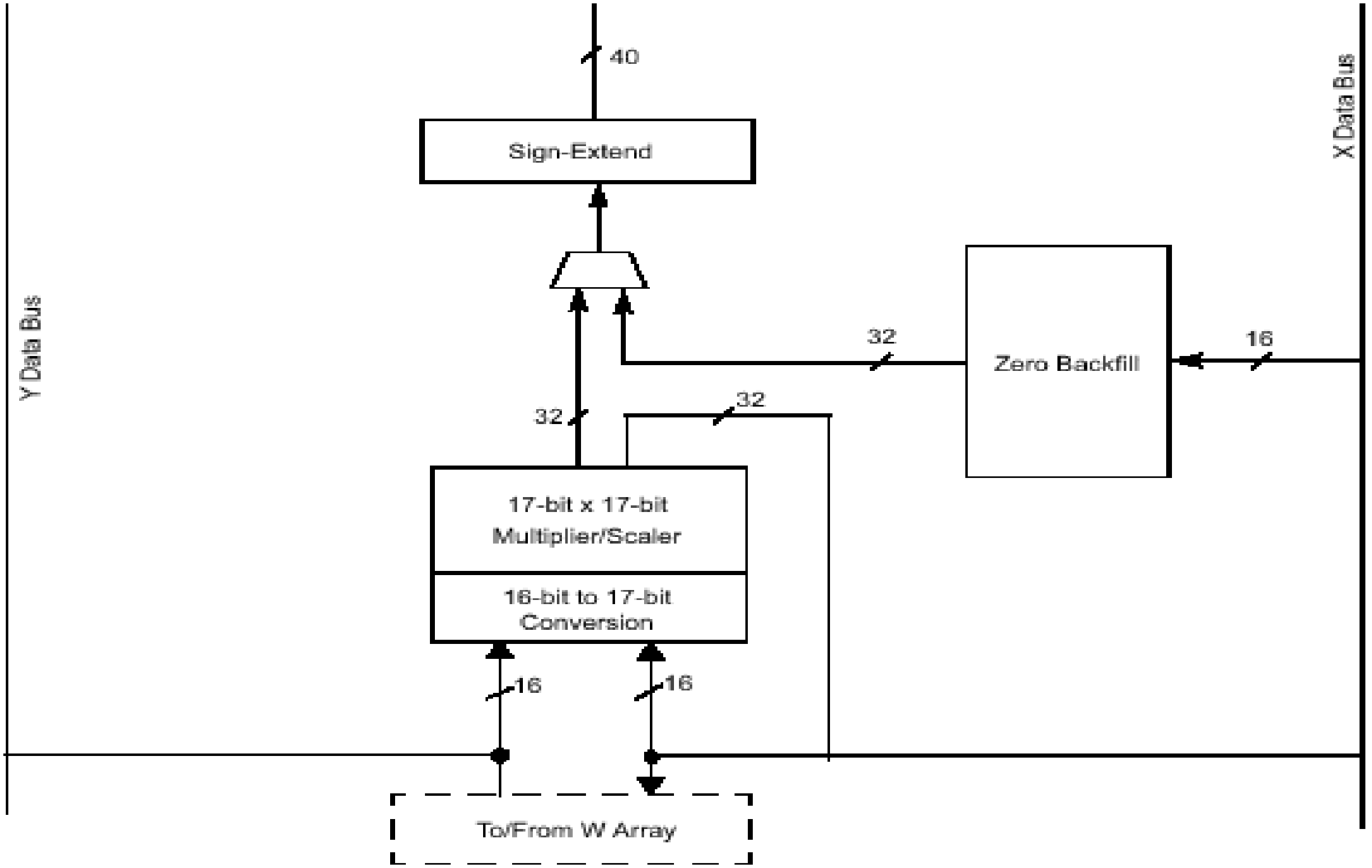
Caractéristiques du 33FJ16MC304

- Coeur 16 bits – 40Mips - 2ko de RAM- 16ko de Flash
- Alimentation : 3v à 3.6 V
- Oscillateurs internes **FRC=7.37MHz** ou LPRC=32.768kHz
- Oscillateur externe jusqu'à 40 MHz
- PLL et diviseur d'horloge intégrés configurable
- Fréquence cycle (instruction, Timers,etc..) **Fcy = Fosc /2**
- 9 ports d ’entrées/sorties 8bits
- 9 entrées analogiques 12bits- 500kech/s ou 10bits-1100kech/s
- 3 Timers 16 bits , 1 Timer 32bits
- Sorties comparaison et PWM (10)
- Entrées de capture, QEP
- 1 liaisons série asynchrone SCI (Uart)
- 1 liaisons série synchrone SPI
- 1 bus I2C


X Address Bus
Y Data Bus
X Data Bus
Interrupt Controller
PSV & Table Data Access Control Block
24
16
Data Latch
Y Data RAM (4 Kbytes)
Address Latch
Data Latch
X Data RAM (4 Kbytes)
Address Latch
PCU
PCH
PCL
Program Counter
Stack Control Logic
Loop Control Logic
Y Address Bus
X RAGU
X WAGU
Y AGU
Address Latch
Program Memory (144 Kbytes)
Data EEPROM (4 Kbytes)
Data Latch
EA MUX
ROM Latch
IR
Literal Data
16 x 16
W Reg Array
Instruction Decode & Control
Control Signals to Various Blocks
Power-up Timer
DSP Engine
Divide Support


(4 Kbytes)
Data Latch
16
24
ROM Latch
IR
16
EA MUX
16
16
Literal Data
16
16 x 16
W Reg Array
16
16
16
Instruction
Decode &
Control
Control Signals
to Various Blocks
DSP
Engine
Divide
Support
16-bit ALU
16
16
Power-up
Timer
Oscillator
Start-up Timer
POR/BOR
Reset
Watchdog
Timer
Low Voltage
Detect
OSC1/CLKI
Timing
Generation
MCLR
V, V
AV, AV
CAN1,
CAN2
12-bit ADC
Input
Capture
Module
Output
Compare
Module
I²C
I/O Ports
Timers
DCI
SPI1,
SPI2
UART1,
UART2


40
40-bit Accumulator A
40-bit Accumulator B
Saturate
Adder
Negate
40
40
40
40
Round Logic
Saturation Logic
16
Barrel
Shifter
16
40
Sign-Extend
X Data Bus


Y Data Bus
X Data Bus
40
Sign-Extend
32
Zero Backfill
16
32
32
17-bit x 17-bit
Multiplier/Scaler
16-bit to 17-bit
Conversion
16
16
To/From W Array

TROIS TYPES DE MEMOIRES

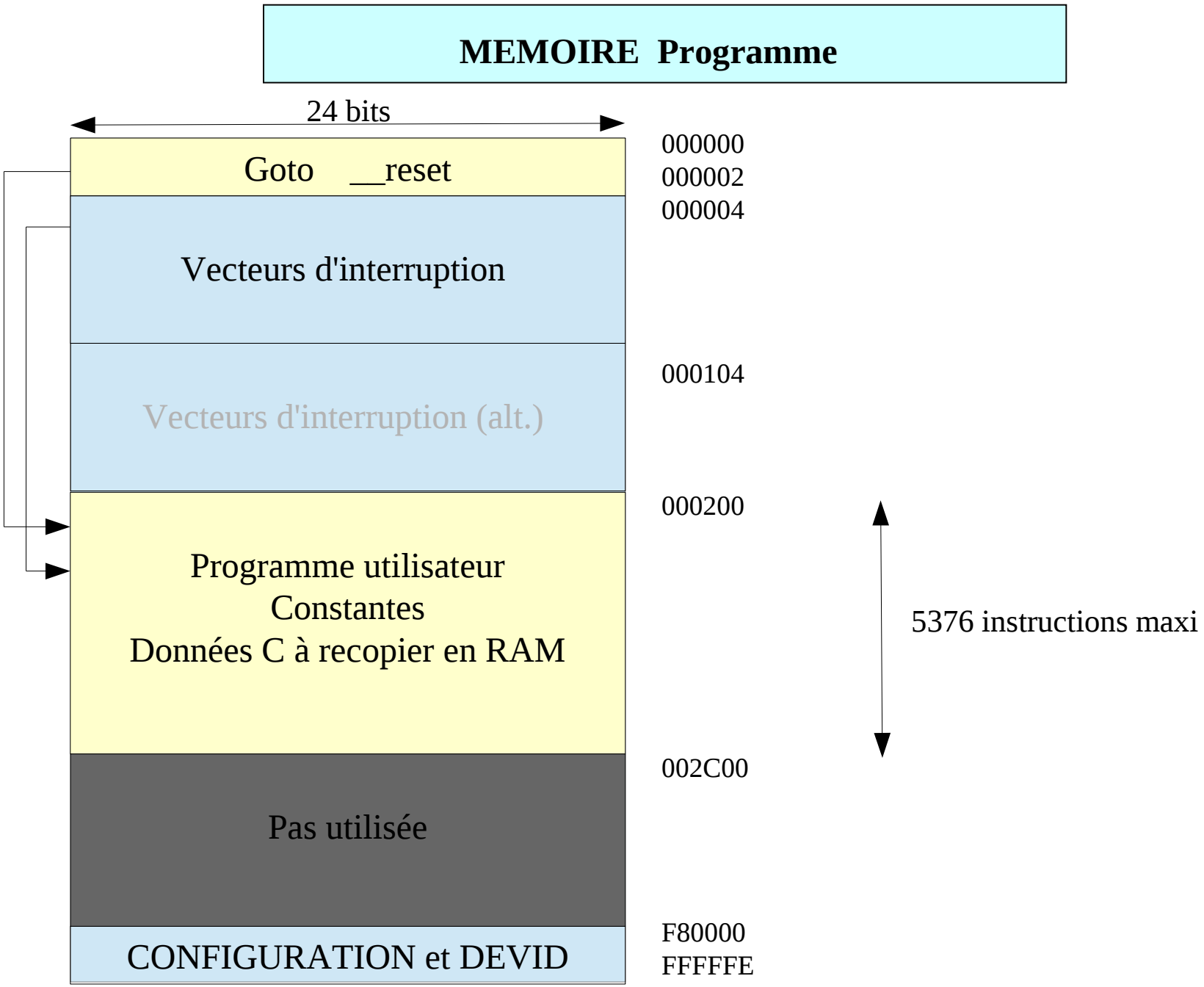
Mémoire programme:

- flash 48 Kmots de 24 bits (144 k octets)

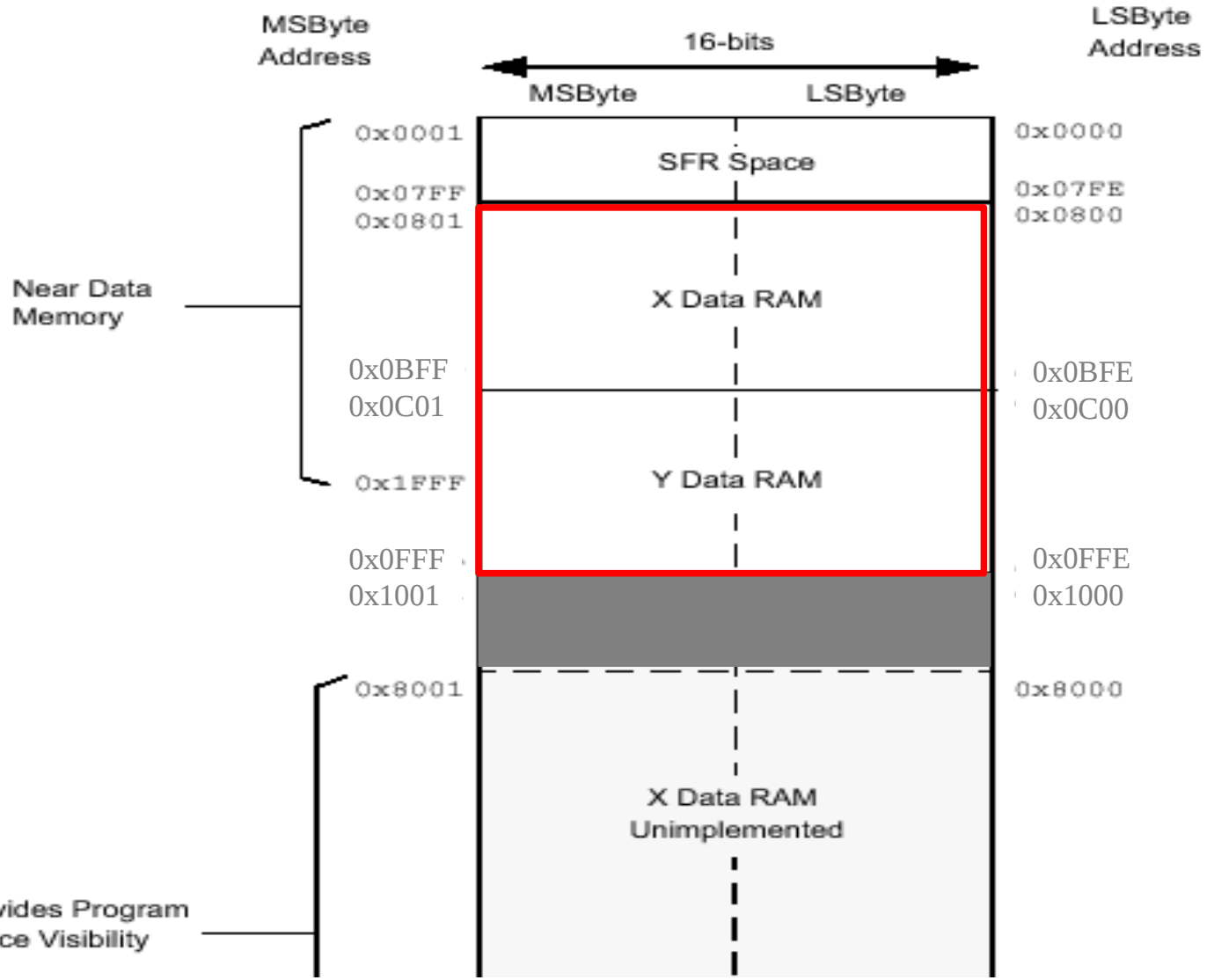
Mémoire données en EEPROM : 8k octets

Mémoire données RAM: 32 Kmots de 16 bits (64 koctets)

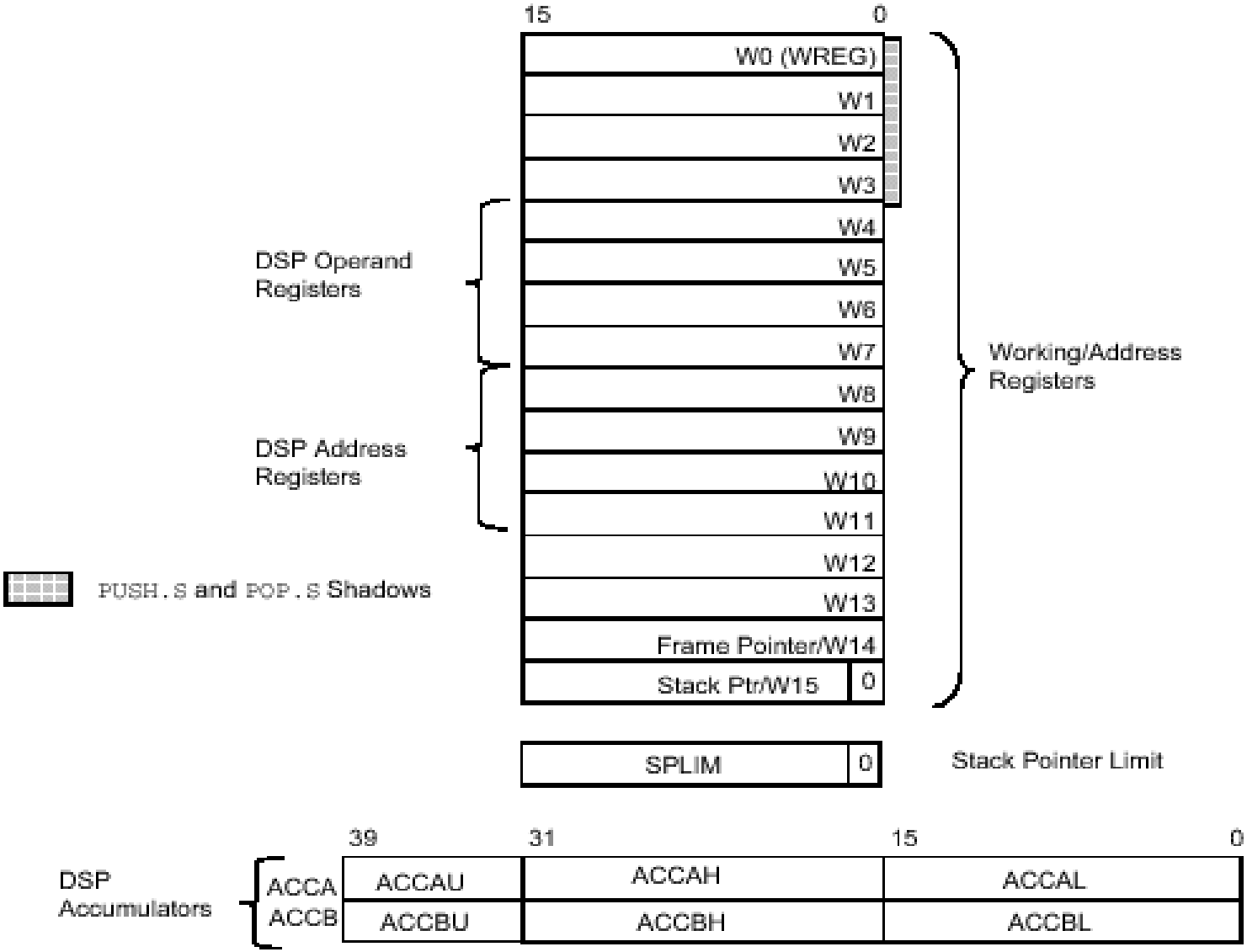
- Les registres système et périphériques internes (SFR) 1k mots (2koctets)
- RAM X 2k mots (4k octets)
- RAM Y 2k mots (4k octets)
- les derniers 16 k mots peuvent être utilisés pour lire la ROM

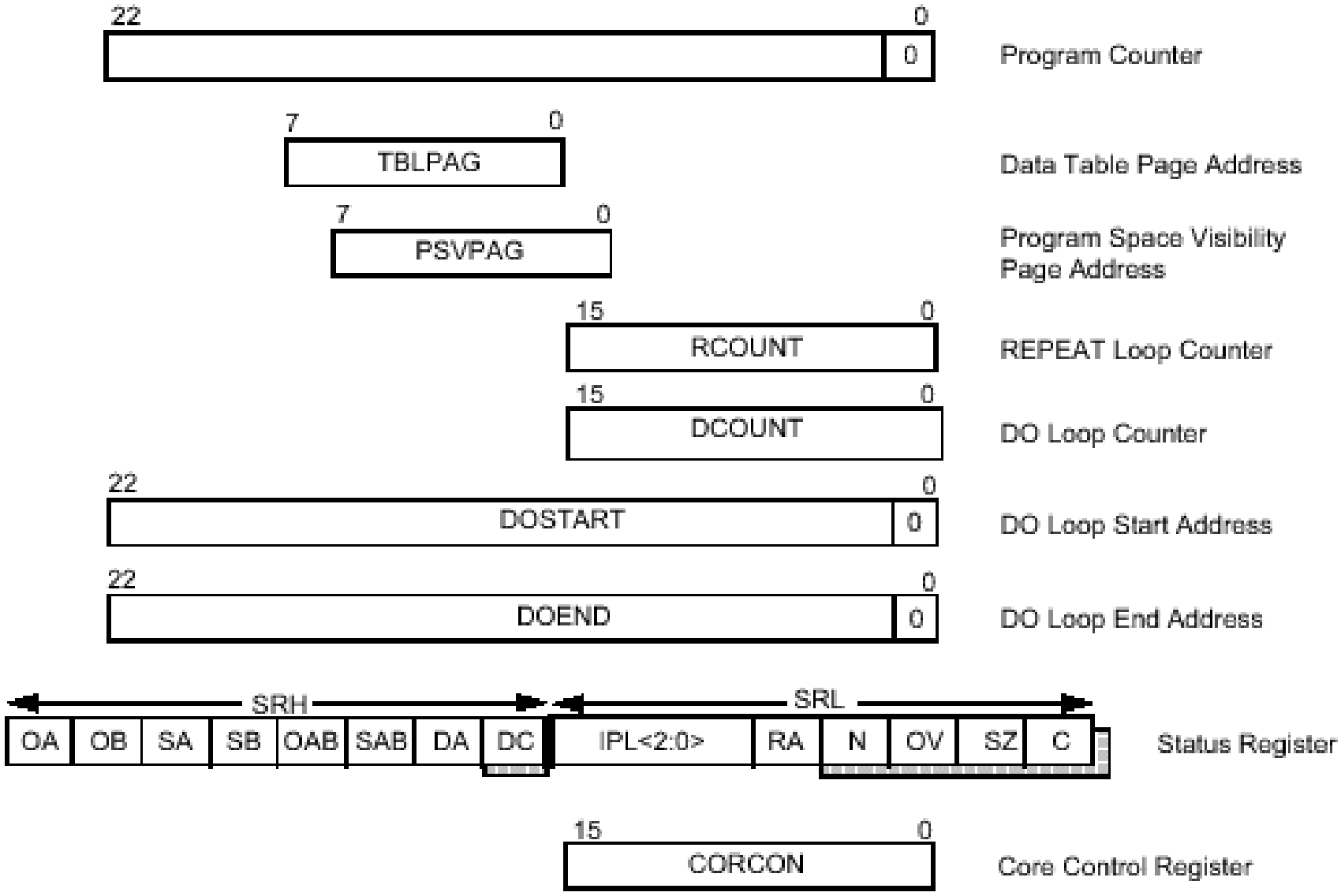

MEMOIRE Programme
24 bits
Goto __reset
Vecteurs d'interruption
Vecteurs d'interruption (alt.)
Programme utilisateur
Constantes
Données C à recopier en RAM
Pas utilisée
CONFIGURATION et DEVID
000000
000002
000004
000104
000200
002C00
F80000
FFFFFE
5376 instructions maxi

MEMOIRE DONNEES



LES PRINCIPAUX REGISTRES




22
0
0
Program Counter
7
0
TBLPAG
Data Table Page Address
7
0
PSVPAG
Program Space Visibility
Page Address
15
0
RCOUNT
REPEAT Loop Counter
15
0
DCOUNT
DO Loop Counter
22
0
DOSTART
0
DO Loop Start Address
22
0
DOEND
0
DO Loop End Address
SRH
SRL
OA
OB
SA
SB
OAB
SAB
DA
DC
IPL<2:0>
RA
N
OV
SZ
C
Status Register
15
0
CORCON
Core Control Register

LES PRINCIPAUX MODES D ’ADRESSAGE

Une Instruction occupe 24 bits dans la mémoire programme

- **Adressage IMMEDIAT:**
  La donnée se trouve à la suite de l ’instruction dans la mémoire FLASH

- **Adressage DIRECT**
  L ’instruction contient l ’adresse sur 13bits) de la donnée qui se trouve en RAM (accès limité à 8k adresses)

- **Adressage indirect**
  Un des registres de travail Wn contient l ’adresse de la donnée qui se trouve en RAM
  - post-incrémentation et post-décrémentation
  - pre-incrémentation et pre-décrémentation
  - offset

- **Adressage « bit »**
  - permet de modifier ou de tester un bit d ’un registre en mémoire donnée

ADRESSAGE IMMEDIAT

Symbole #

Mnémonique #nombre , destination

- Exemples :

```
mov   #0x01C2, W3

mov.b  #0x05, W1

add  #0x05 , W0
```

- Autres instructions : sub, and , ior, xor…

- Taille du #nombre :

  #lit10 signifie « literal » de 10 bits maxi….

ADRESSAGE DIRECT

Mnémonique adresse , destination
Mnémonique source , adresse

- Exemples :

```
mov   0x01C2, W3

mov.b  W1,  0x0005

add  0x0105 , W0

rlc 0x01F0 , W0

rlc 0x01F0
```

- Autres instructions : rlnc, asl, ls, rrc, sub, and , ior, xor…

ADRESSAGE INDIRECT

Mnémonique [source] , destination
Mnémonique source , [destination]
Mnémonique [source] , [destination]

[Wx] Utiliser le contenu de l ’adresse Wx

[Wx ++] et [Wx - -] Utiliser le contenu de l ’adresse Wx puis ajouter ou retirer 1 à Wx

[++Wx], [- -Wx] Ajouter ou retirer 1 à Wx puis **u**tiliser le contenu de l ’adresse Wx

[Wx+Wy] Utiliser le contenu de l ’adresse calculée

- Exemples :

```
mov   [W2] , W3
mov   [--W2] , W3
mov    W5 , [W1+W0]
mov  [W5--] , [++W2]

add   Wx , [Wy++] , [Wz]
```

ADRESSAGE « bit »

Mnémonique destination, #numéro du bit

Exemples:

```
bset W6, #0

bclr  0x105A, #7

btsc  [W5], #2

btss  0x105A, #3
```

Sauts et sauts conditionnels

Mnémonique (condition ,) adresse-programme

Label ou étiquette = adresse de la mémoire programme

- Saut simple

```
goto  label1
bra   label2
```

- Saut conditionnel

Z, C, N, GE, GT, LE,LT,OA,….

Ex: bra z, label3

Appels de sous-programmes

APPEL :

- *CALL adresse-programme (ou label)*

  - L ’adresse de retour est sauvegardée dans la pile
  - W15 contient l ’adresse du sommet de la pile

RETOUR :

- *RETURN* ou *RETLW #nombre,Wx*

  - L ’adresse de retour est « dépilée »

Les instructions dédiées au traitement du signal

- Les modes de fonctionnement du multiplicateur (registre **CORCON**)
  - Signed/Unsigned
  - Saturation/Supersaturation (32 ou 40 bits)
  - Saturation enable/disable
  - Integer/Fractionnal (décalage du résultat pour format Q31)

- Les instructions:

  CLR, MPY, MAC, MSC, MOVSAC, LAC, SAC,SFTAC

- Fonctionnement du multiplicateur (MPY , MAC, MSC)
  - Les grandeurs à multiplier doivent être dans ***2 registres (W4…W7)***
  - Le résultat est stocké, ajouté ou soustrait à un accumulateurs (A ou B).
  - Le contenu des 2 registres peut être modifié, pour la multiplication suivante, grâce à l ’adressage indirect
    - . de la RAM X par : [W8] ou [W9] +/-2, 4 6
    - . de la RAM Y par : [W10] ou [W11] +/-2, 4 6
  - Les bits 16 à 31 du deuxième accumulateur peuvent-être transférés dans [W13]

Fonctionnement du multiplicateur (MPY , MAC, MSC)

- Les grandeurs à multiplier doivent être dans ***2 registres (W4…W7)***

- Le résultat est stocké, ajouté ou soustrait à un accumulateurs (A ou B).

- Le contenu des 2 registres peut être modifié, pour la multiplication suivante, grâce à l ’adressage indirect

  . de la RAM X par : [W8] ou [W9] +/-2, 4 6

  . de la RAM Y par : [W10] ou [W11] +/-2, 4 6

- Les bits 16 à 31 du deuxième accumulateur peuvent être transférés dans [W13]

Acc1 = Wm*Wn

Wm = [Wx]

Wn = [Wy]

MPY

Multiply Wm by Wn to Accumulator

Syntax:	{label:}	MPY	Wm*Wn, Acc	{,[Wx], Wxd}	{,[Wy], Wyd}
				{,[Wx]+=kx, Wxd}	{,[Wy]+=ky, Wyd}
				{,[Wx]-=kx, Wxd}	{,[Wy]-=ky, Wyd}
				{,[W9+W12], Wxd}	{,[W11+W12], Wyd}

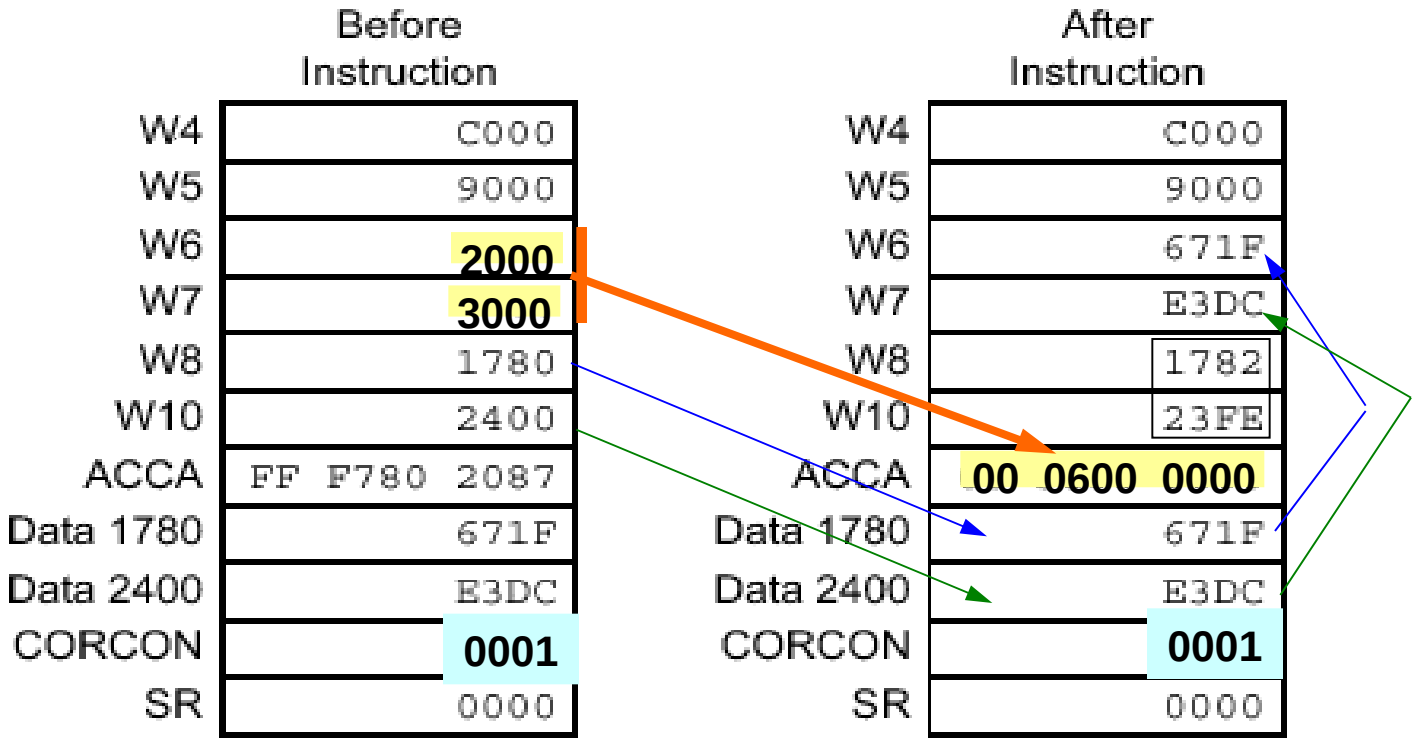
Operands: Wm*Wn ∈ [W4*W5, W4*W6, W4*W7, W5*W6, W5*W7, W6*W7]
Acc ∈ [A,B]
Wx ∈ [W8, W9]; kx ∈ [-6, -4, -2, 2, 4, 6]; Wxd ∈ [W4 ... W7]
Wy ∈ [W10, W11]; ky ∈ [-6, -4, -2, 2, 4, 6]; Wyd ∈ [W4 ... W7]
AWB ∈ [W13, [W13]+=2]

Operation: (Wm)*(Wn) → Acc(A or B)
([Wx])→ Wxd; (Wx)+kx→Wx
([Wy])→ Wyd; (Wy)+ky→Wy

Status Affected: OA, OB, OAB, SA, SB, SAB

Exemple :

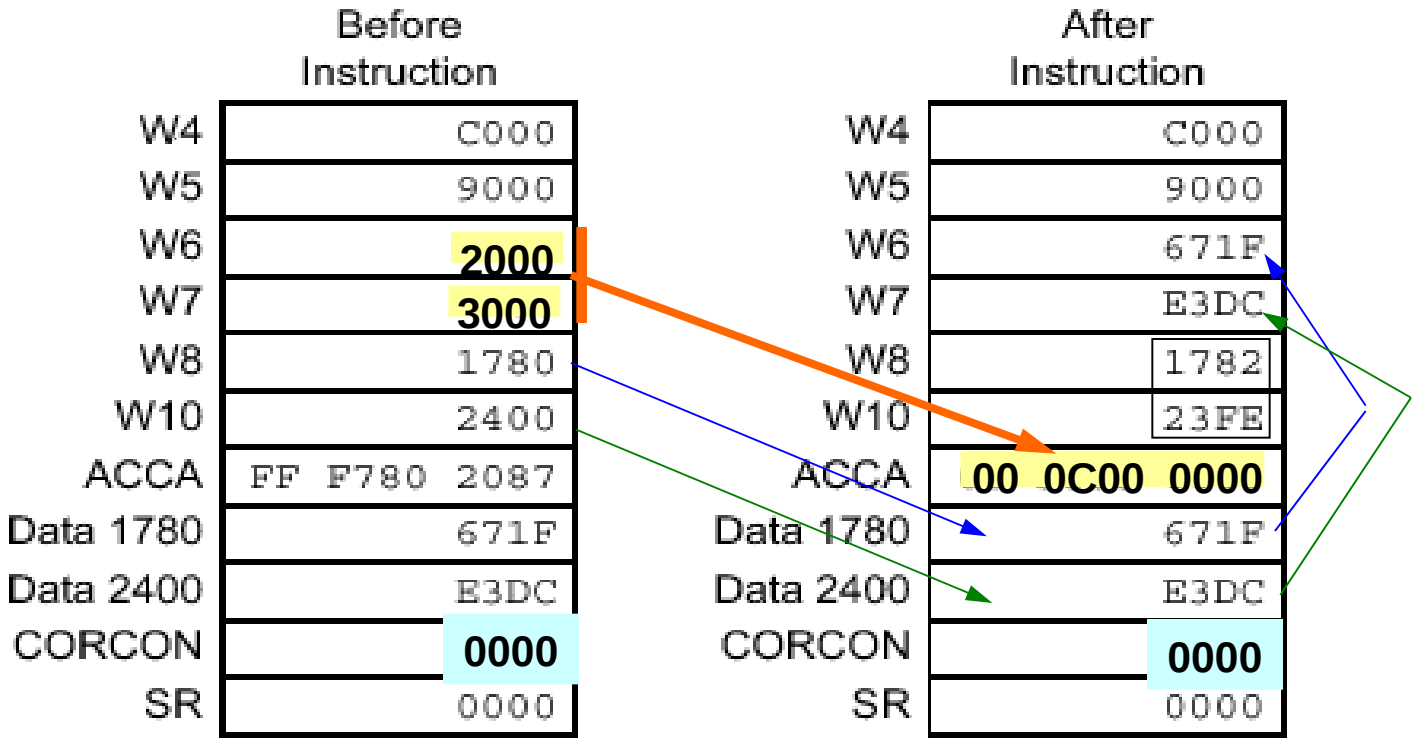
MPY W6*W7, A, [W8]+=2, W6, [W10] - =2, W7



Exemple :

MPY W6*W7, A, [W8]+=2, W6, [W10] - =2, W7

$$A = W6*W7$$
$$W6 = [W8] \quad ; \quad W8=W8+2$$
$$W7 = [W10] \; ; \; W10=W10-2$$



Acc1 = Acc1 + Wm*Wn

Wm = [Wx]

Wn = [Wy]

W13 ou [W13] = Acc2

MAC

Multiply and Accumulate

Syntax:	{label:}	MAC	Wm*Wn, Acc	{,[Wx], Wxd}	{,[Wy], Wyd}	{,AWB}
				{,[Wx]+=kx, Wxd}	{,[Wy]+=ky, Wyd}	
				{,[Wx]-=kx, Wxd}	{,[Wy]-=ky, Wyd}	
				{,[W9+W12], Wxd}	{,[W11+W12], Wyd}	

Operands:

Wm*Wn ∈ [W4*W5, W4*W6, W4*W7, W5*W6, W5*W7, W6*W7]
Acc ∈ [A,B]
Wx ∈ [W8, W9]; kx ∈ [-6, -4, -2, 2, 4, 6]; Wxd ∈ [W4 ... W7]
Wy ∈ [W10, W11]; ky ∈ [-6, -4, -2, 2, 4, 6]; Wyd ∈ [W4 ... W7]
AWB ∈ [W13, [W13]+=2]

Operation:

(Acc(A or B)) + (Wm)*(Wn) → Acc(A or B)
([Wx])→ Wxd; (Wx)+kx→Wx
([Wy])→ Wyd; (Wy)+ky→Wy
(Acc(B or A)) rounded → AWB

Status Affected:

OA, OB, OAB, SA, SB, SAB

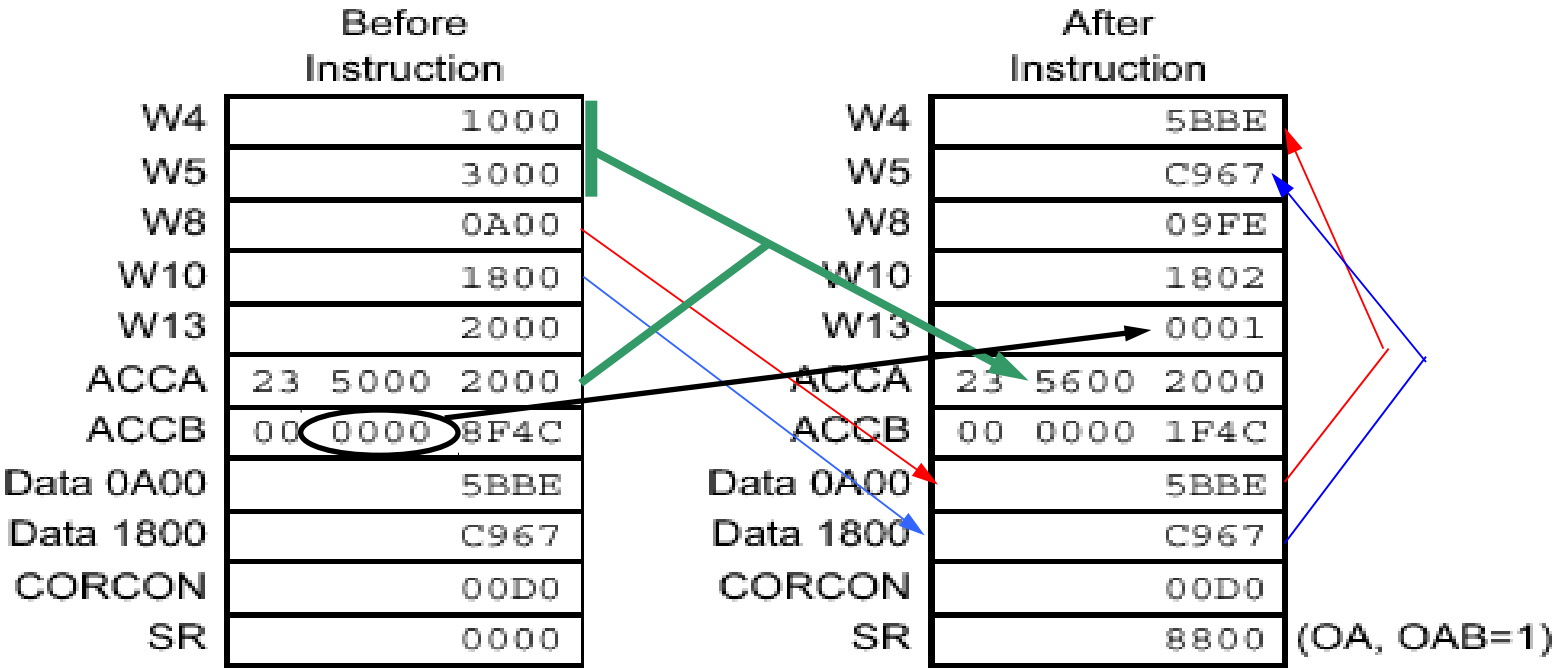
Exemple :

MAC W4*W5, A, [W8]-=2, W4, [W10] + =2, W5, W13

$$A = A + W4*W5$$

$$W4 = [W8] \;\; ; \;\; W8=W8 - 2$$

$$W5 = [W10] \; ; \; W10=W10 + 2$$



Les buffers circulaires

Principe :

- On initialise les registres de contrôle
- On met l ’adresse de départ dans Wx
- Wx++ incrémente Wx
- lorsque Wx atteint la fin du buffer il revient automatiquement à l ’adresse de départ

Registres de contrôle :

MODCON :

- Autorisations des buffers X et Y
- Choix des registres pointeurs Wx et Wy (W0 …. W14)

XMODSRT et YMODSRT : adresses de départ

XMODEND et YMODEND : adresses de fin

XBREV et YBREV : pour l ’adressage ‘ bit reverse ’

Utilisations des Timers_

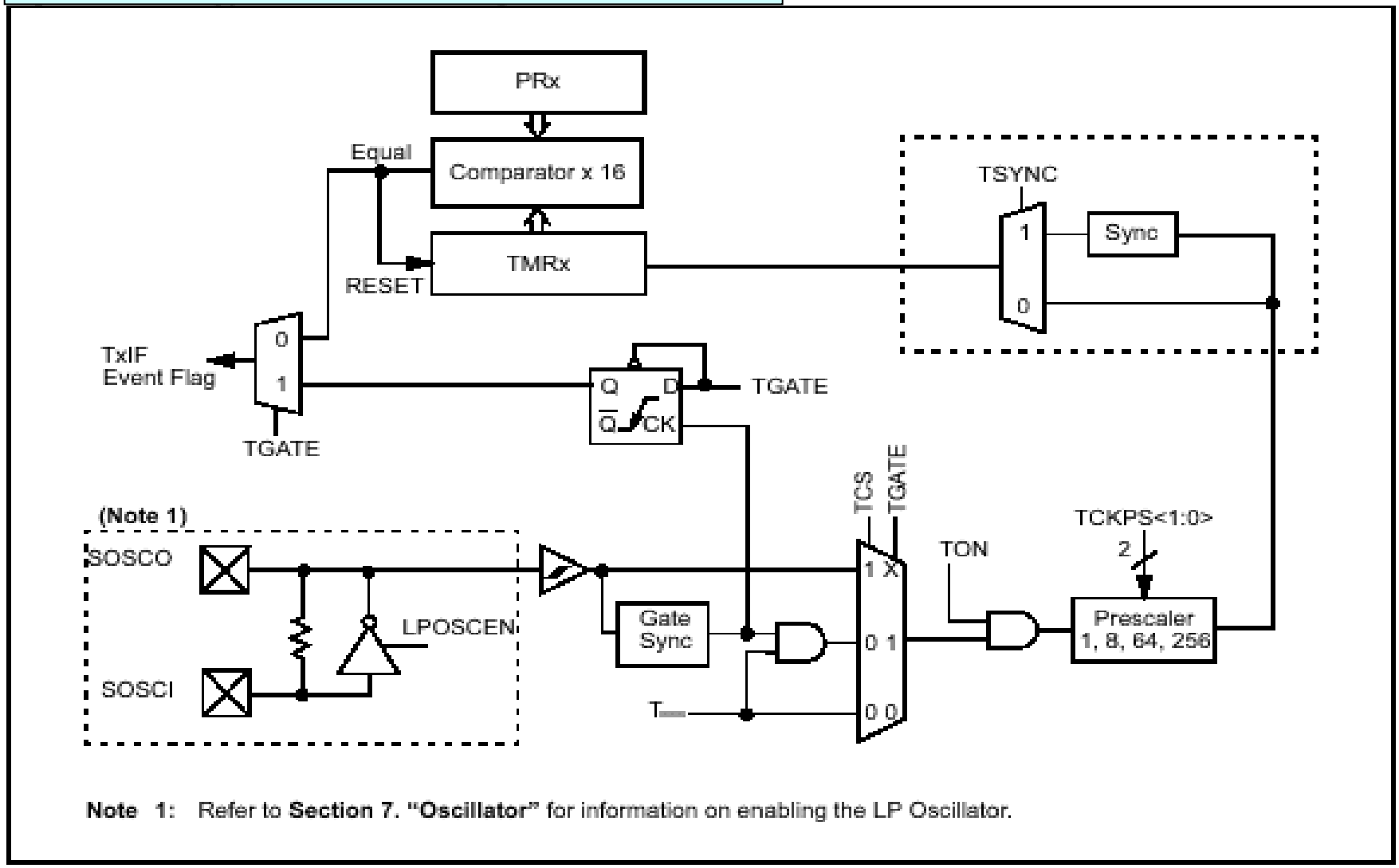
5 **Timers indépendantes**:

Génération de période d'échantillonnage
Bases de temps pour mesure de durée (Capture des Timers)
Compteur d 'impulsions extérieures
Gestion de codeurs incrémentaux (QEP)
Bases de temps pour la génération automatique des signaux PWM
Gestion d 'interruptions sur événements Timers, capture, comparaison...

Registres associés à un timer

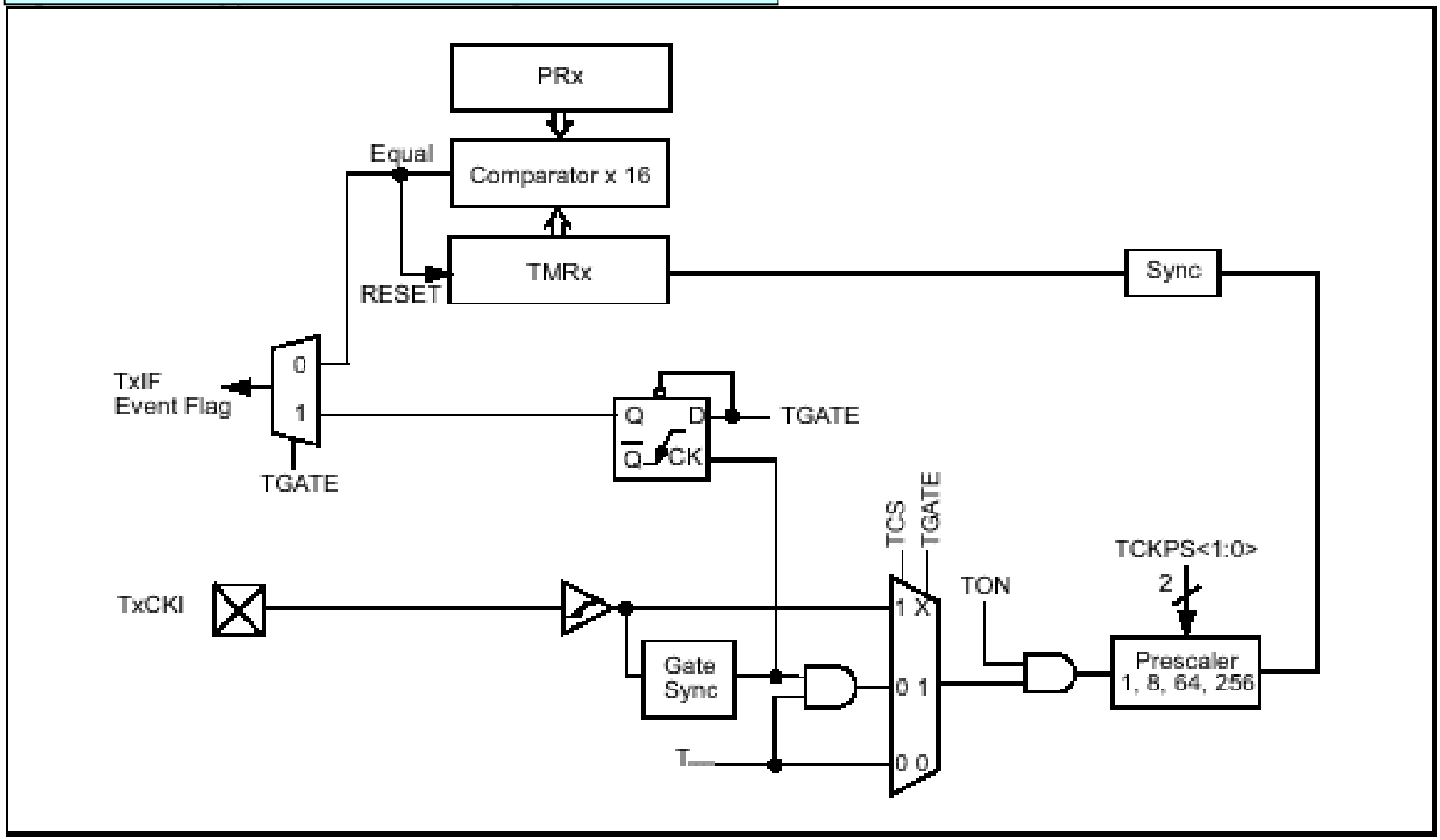
un registre compteur	TMRx
un registre de contrôle	TxCON
un registre période	PRx

TIMER 1

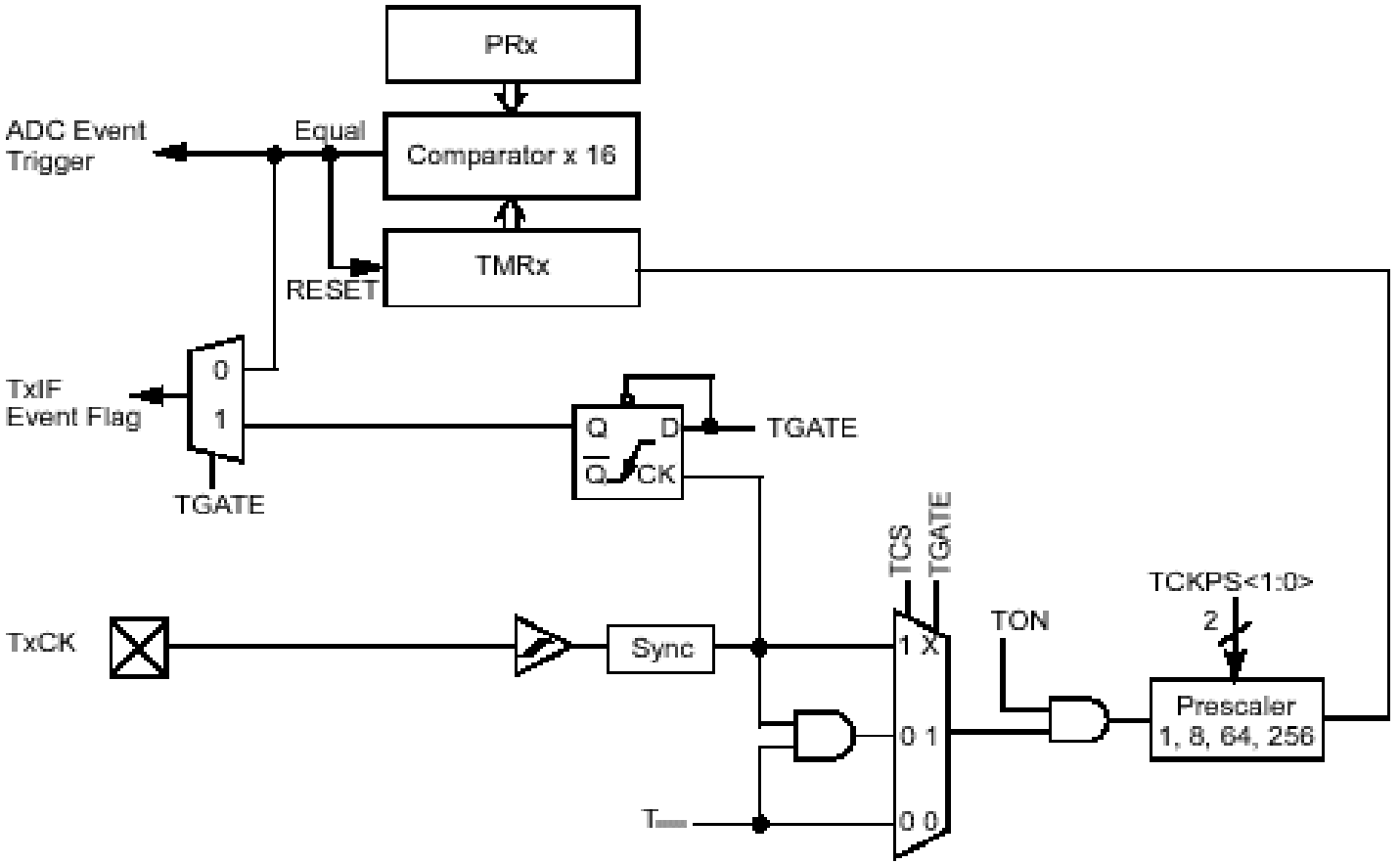


Note 1: Refer to **Section 7. "Oscillator"** for information on enabling the LP Oscillator.

TIMER 2

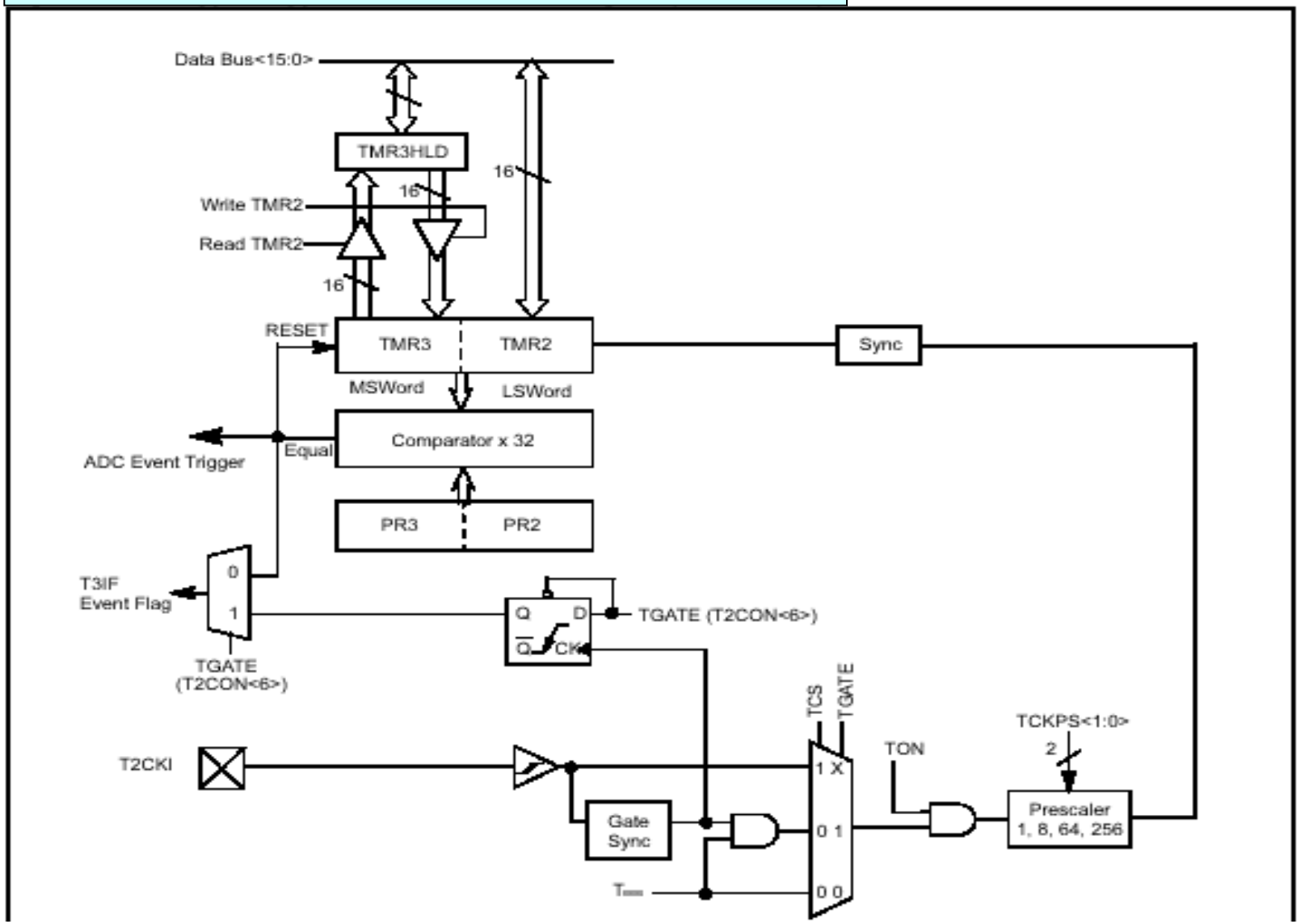

PRx
Equal
Comparator x 16
TMRx
RESET
Sync
TxIF
Event Flag
0
1
TGATE
Q
D
Q
CK
TGATE
TCS
TGATE
TCKPS<1:0>
TON
2
TxCKI
1 X
Gate
Sync
0 1
T
0 0
Prescaler
1, 8, 64, 256

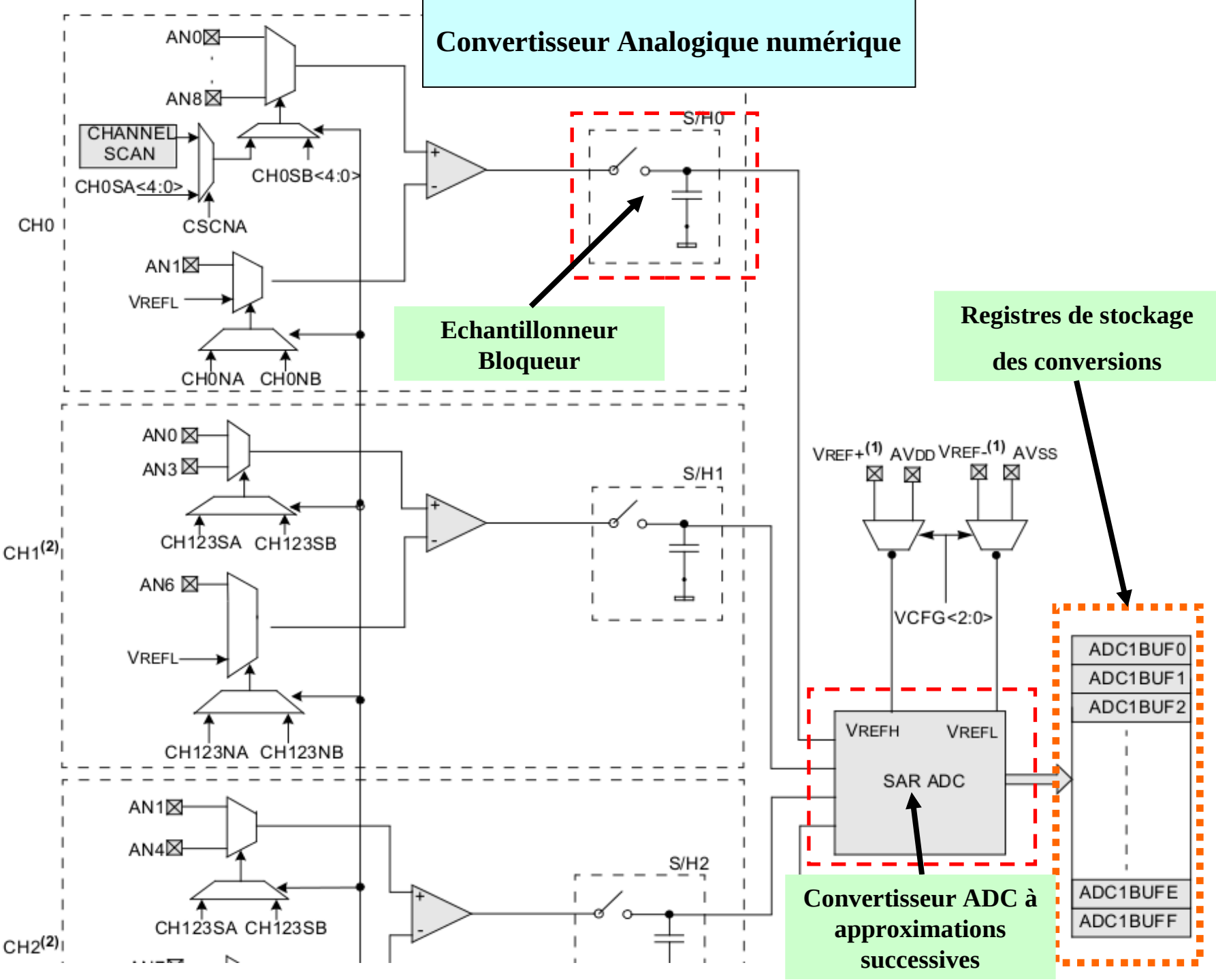
TIMER 3



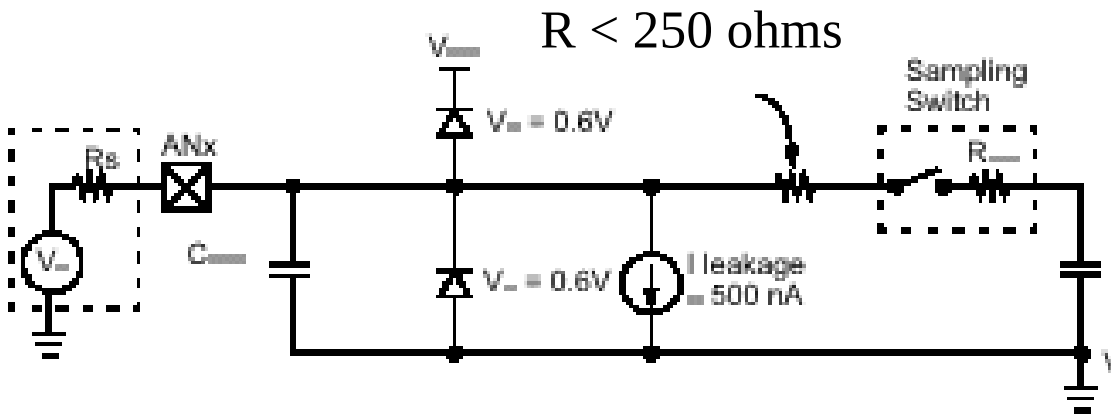
Note: In certain variants of the dsPIC30F family, the TxCK pin may not be available. Refer to the device data sheet for the I/O pin details. In such cases, the timer must use the system clock (F/4) as its input clock, unless it is configured for 32-bit operation.

TIMER 2+3 (32bits)

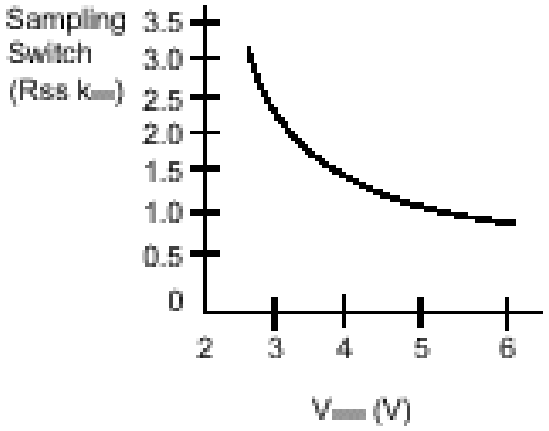



Convertisseur Analogique numérique
AN0
AN8
CHANNEL SCAN
CH0SA<4:0>
CH0SB<4:0>
CSCNA
CH0
AN1
VREFL
CH0NA
CH0NB
S/H0
Echantillonneur Bloqueur
Registres de stockage des conversions
AN0
AN3
CH123SA
CH123SB
CH1(2)
S/H1
AN6
VREFL
CH123NA
CH123NB
VREF+(1)
AVDD
VREF-(1)
AVSS
VCFG<2:0>
ADC1BUF0
ADC1BUF1
ADC1BUF2
VREFH
VREFL
SAR ADC
AN1
AN4
CH123SA
CH123SB
CH2(2)
S/H2
ADC1BUFE
ADC1BUFF
Convertisseur ADC à approximations successives

Caractéristiques électriques d ’une entrée analogique



Legend:
- C = input capacitance (approximately 5 pF)
- V = threshold voltage
- I leakage = leakage current at the pin due to various junctions
- R = interconnect resistance
- SS = sampling switch
- C = sample/hold capacitance (from DAC)



Note: C value depends on device package and is not tested. Effect of C negligible if Rs ≤ 2.5 kΩ

Temps d ’échantillonnage et de conversion

- Echantillonnage:

$$T_{SMP} = 0.5\ \mu s + C_{HOLD} \bullet (R_{IC} + R_{SS} + R_S) \bullet \ln(2 \bullet 4095)$$

- Période d ’horloge du convertisseur:

T_{AD} mini = 667ns

$$T_{AD} = T_{CY} * (0.5*(ADCS<5:0> + 1))$$

- Temps de conversion :

10 bits = $12*T_{AD}$

12 bits = $14 * T_{AD}$

Calcul du résultats d ’une conversion

- Tension d ’entrée Ve $\quad Vref- < Ve < Vref+$

  - *Vref+ maxi = 5V*
  - *Vref- mini = 0*

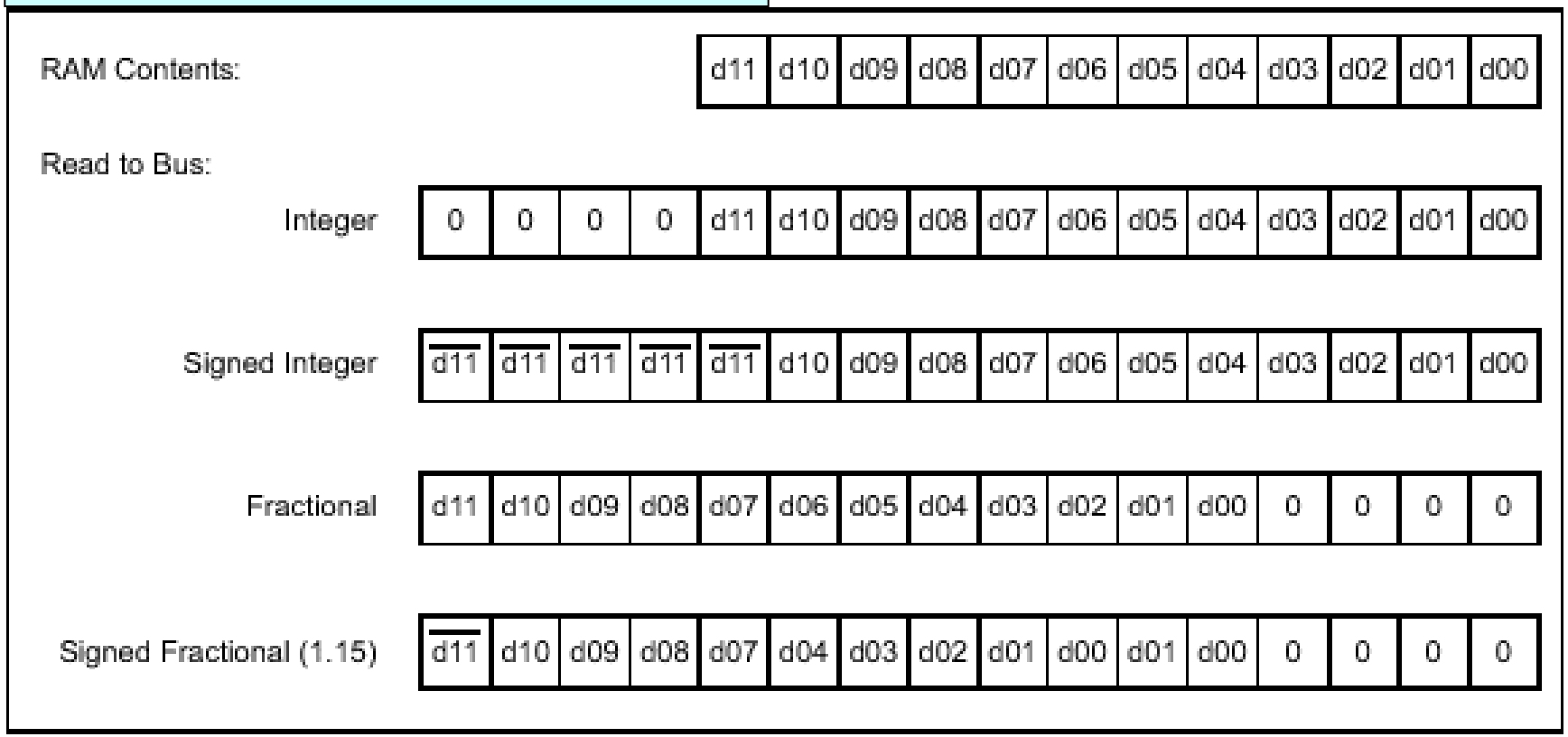
- Résultat **n** sur 12 bits $\quad 0 < n < 2^{12} - 1$

$$n = round(4095 * (Ve - Vref_-)/ (Vref_+ - Vref_-))$$

- Résolution : $1\ lsb = (Vref_+ - Vref_-)/4095$

- La Précision dépend de Ve

Formats du résultat de la conversion



Les registres du convertisseur Analogique Digital

- **AD1CON1 Format du résultat, déclenchement,..**
- **AD1CON2 Choix Vref+/-, mode de fonctionnement..**
- **AD1CON3 Paramétrage de l ’horloge de conversion**
- **AD1CHS0 ou 123 Choix des entrées, reférences...**
- **AD1PCFGL Configuration des broches du circuit**
- **AD1CSSL Choix des entrées à balayer (scan)**
- **ADCBUF0 ……ADCBUFF Registres résultats**

Utilisation du convertisseur Analogique Digital

INITIALISATION

- Configurer les entrées (analogiques ou numériques)
- Choisir les tensions de référence Vref+ et Vref-
  les entrées à convertir,
  l 'horloge de conversion,
  le mode de déclenchement
- Activer le convertisseur

. Configurer les interruptions si nécessaire

Traitement par POLLING:

. Lancer la conversion (si pas automatique)
. Attendre la fin de conversion
. Lire le résultat

Traitement par routine d 'interruption ADC

. La routine est appelée automatiquement en fin de conversion
. Lire le résultat et réinitialiser l 'indicateur d 'interruption
(La conversion peu être lancée automatiquement par le timer3 ou par le programme principal)

Les sources d ’interruptions

- 8 non masquables de priorité fixe 8 à 15 (hard et soft)

- 118 masquables de priorité configurable (périphériques)

- Registres INTCON1 , INTCON2

- Registres pour autoriser/interdire IEC0, IEC1 et IEC2 (1 pour autoriser)

- Registres indicateurs IFS0, IFS1 et IFS2 (mis à 1 lors d ’une demande d ’int.)
  (à remettre à 0 par soft)

- Registre pour gérer les priorités IPC0….IPC10 (0 faible …7 élevée)

- Registres également concernés (INTREG, SR et CORCON)

Table 6-1: Trap Vector Details

Vector Number	IVT Address	AIVT Address	Trap Source
0	0x000004	0x000084	Reserved
1	0x000006	0x000086	Oscillator Failure
2	0x000008	0x000088	Address Error
3	0x00000A	0x00008A	Stack Error
4	0x00000C	0x00008C	Arithmetic Error
5	0x00000E	0x00008E	Reserved
6	0x000010	0x000090	Reserved
7	0x000012	0x000092	Reserved

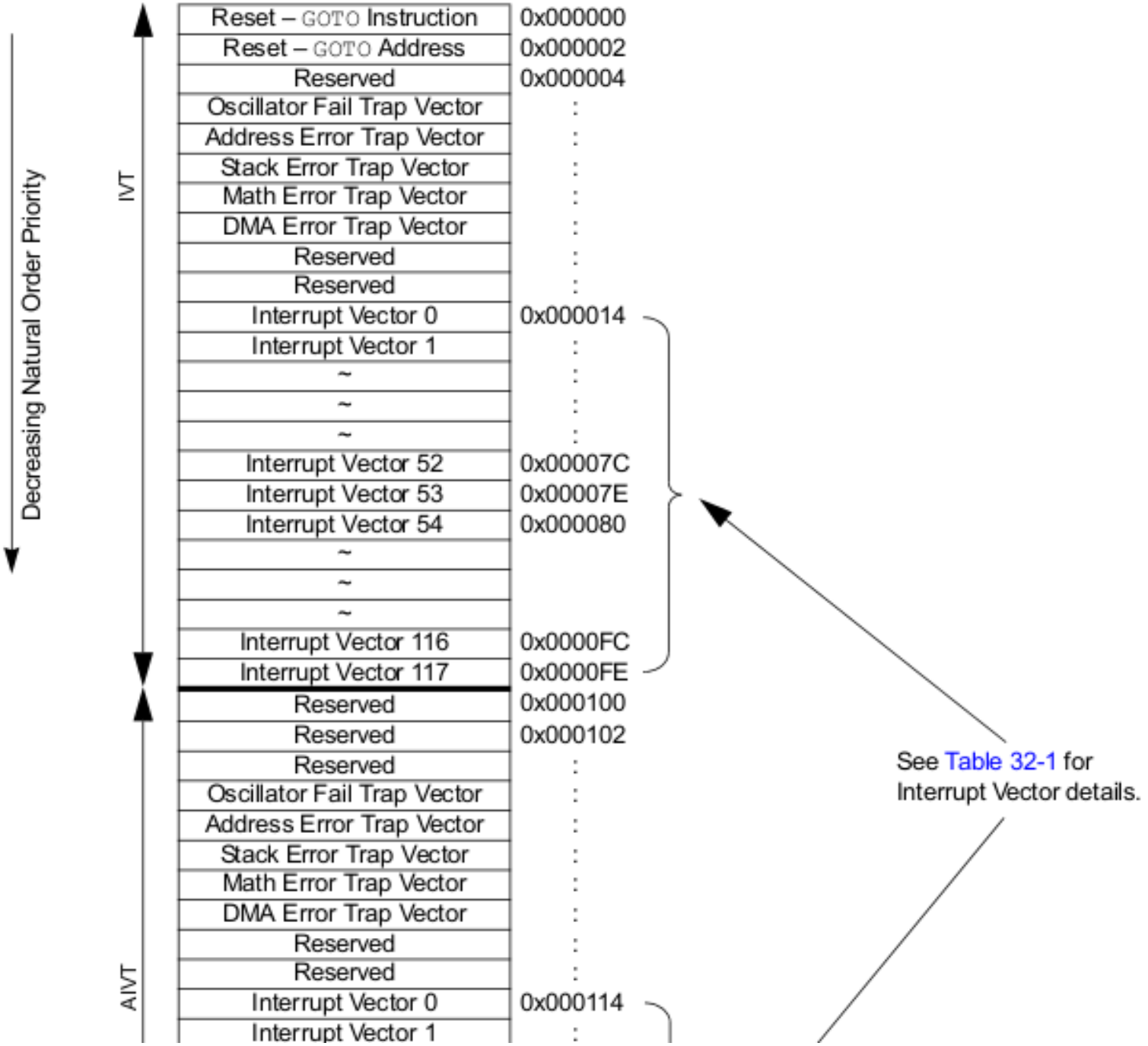

Decreasing Natural Order Priority
IVT
AIVT
Reset – GOTO Instruction
Reset – GOTO Address
Reserved
Oscillator Fail Trap Vector
Address Error Trap Vector
Stack Error Trap Vector
Math Error Trap Vector
DMA Error Trap Vector
Reserved
Reserved
Interrupt Vector 0
Interrupt Vector 1
~
~
~
Interrupt Vector 52
Interrupt Vector 53
Interrupt Vector 54
~
~
~
Interrupt Vector 116
Interrupt Vector 117
Reserved
Reserved
Reserved
Oscillator Fail Trap Vector
Address Error Trap Vector
Stack Error Trap Vector
Math Error Trap Vector
DMA Error Trap Vector
Reserved
Reserved
Interrupt Vector 0
Interrupt Vector 1
0x000000
0x000002
0x000004
0x000014
0x00007C
0x00007E
0x000080
0x0000FC
0x0000FE
0x000100
0x000102
0x000114
See Table 32-1 for
Interrupt Vector details.

Table 32-1: Interrupt Vectors

Vector Number	IVT Address	AIVT Address	Interrupt Source
0	0x000004	0x000104	Reserved
1	0x000006	0x000106	Oscillator Failure
2	0x000008	0x000108	Address Error
3	0x00000A	0x00010A	Stack Error
4	0x00000C	0x00010C	Math Error
5	0x00000E	0x00010E	Direct Memory Access (DMA) Error
6-7	0x000010-0x000012	0x000110-0x000112	Reserved
8	0x000014	0x000114	External Interrupt 0 (INT0)
9	0x000016	0x000116	Input Capture 1 (IC1)
10	0x000018	0x000118	Output Compare 1 (OC1)
11	0x00001A	0x00011A	Timer1 (T1)
12	0x00001C	0x00011C	DMA Channel 0 (DMA0)
13	0x00001E	0x00011E	Input Capture 2 (IC2)
14	0x000020	0x000120	Output Compare 2 (OC2)
15	0x000022	0x000122	Timer2 (T2)
16	0x000024	0x000124	Timer3 (T3)
17	0x000026	0x000126	SPI1 Error (SPI1E)
18	0x000028	0x000128	SPI1 Transfer Done (SPI1)
19	0x00002A	0x00012A	UART1 Receiver (U1RX)
20	0x00002C	0x00012C	UART1 Transmitter (U1TX)
21	0x00002E	0x00012E	ADC1 Convert Done (AD1)
22	0x000030	0x000130	DMA Channel 1 (DMA1)
23	0x000032	0x000132	Reserved
24	0x000034	0x000134	I2C1 Slave Events (SI2C1)
25	0x000036	0x000136	I2C1 Master Events (MI2C1)
26	0x000038	0x000138	Comparator Interrupt (CMP)

Exécution d'une routine de traitement d ’interruption

- *L ’adresse de retour et les indicateurs sont sauvegardés automatiquement sur la pile*

- Exécution du programme de traitement de l ’interruption
  - Sauvegarde des registres utilisées sur la pile
  - Le traitement…..
  - Ré-initialisation de l ’indicateur d ’interruption concerné
  - Restauration des registres précédemment sauvegardés
  - RETFIE pour retourner au programme interrompu

- *Les indicateurs et l ’adresse de retour sont dépilées automatiquement*

Gestion des priorités

- L ’instruction ***DISI #n*** interdit les interruptions pendant n cycles

- Une interruption peut-être interrompue par un autre interruption de priorité plus élevé (imbrication)

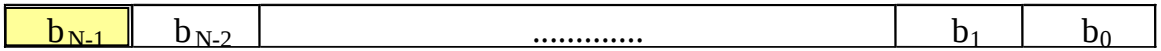
- On peut interdire l ’imbrication (*bit 31 de INTCON1*)

- On peut modifier les priorités (*0 à 7*) des int.

- Priorité 0 = int. interdite

Codage des nombres entiers en complément à 2

Soit X un nombre codé sur N bits en complément à 2



bit de signe :

$b_{N-1} = 0$ si $X > 0$

$b_{N-1} = 1$ si $X < 0$

$$X = -2^{N-1}.b_{N-1} + \sum_{i=0}^{i=N-2} b_i.2^i$$

$$X_{max} = 2^{N-1} - 1$$

$$X_{min} = -2^{N-1}$$

Opération sur des nombres codés en complément à 2

- Addition de 2 nombres de N bits ⇨ Résultat sur N+1 bits au plus (overflow-carry)

- Produit de 2 nombres de N bits ⇨ Résultat sur 2N bits

  Les 2 bits de poids forts du produit sont identiques (2 fois le bit de signe)

  Le MSB est donc inutile.

  Le résultat peut être décalé automatiquement d'un bit à gauche

Représentation des ‘réels ’ en virgule fixe

Le terme « Q_k» indique k bits après la virgule

Si X est codé sur N bits en format Q_k.

Partie entière					Partie fractionnaire			
b_{N-1-K}	b_{N-2-K}	.....	b_1	b_0	b_{-1}	b_{-2}	.....	b_{-K}

$$X = - b_{N-1-K}.2^{N-1-K} + b_{N-2-K}.2^{N-2-K} + ... + b_0 + b_{-1}.2^{-1} + ...+ b_{-K}.2^{-K}$$

- Erreur : $< 2^{-K}$

- Précision : $\underline{2}^{-K} / X$ dépend de X (mauvaise si X est petit…)

-Valeurs limites : $X_{max} = 2^{N-1-K} - 2^{-K}$

$X_{min} = - 2^{N-1-K}$

$| \Delta X_{min} | = 2^{-K}$

Calculs en virgule fixe

Nombres codés dans le même format Qk

- On multiplie X par 2^K.

- Cela correspond à la représentation de l'entier $Y = \text{round}(X.2^K)$

- Les calculs se font alors exactement comme sur les entiers

Exemple de calcul en Q15

Soit la multiplication suivante: a * b

Avec: a = 0.2 Format Q15

b = 0.5 Format Q15

- **On multiplie a et b par 2^{15}**

a = 0.2 x 2^{15} = 6553,6d ≡ 6554d = 199Ah

b = 0.5 x 2^{15} = 16384d = 4000h

- **On multiplie les entiers obtenus**

199Ah x 4000h = 666 8000h au format Q30

- **Résultat mis en Q15 par** :

- Décalage à gauche de 1 (en Q31)
- et copie des 16 bits de poids forts

666 8000h = 0000 0110 0110 0110 1000 0000 0000 0000 b

Après décalage : 0000 1100 1100 1101b = 0CCDh = 3277

- On divise par 2^{15} pour vérifier : 3277/ 2^{15} = 0.100006

- On constate une erreur d ’arrondi de 6.10^{-6}

Calculs en virgule fixe (suite)

Nombres codés dans des formats différents Qk1 et Qk2

Addition: Il faut aligner les virgules en évitant les débordement
Donc convertir deux nombres dans le même format

Multiplication:

Le produit de 2 nombres sur N bits donne un résultat sur 2N bits
Les 2 bits de poids forts sont identiques (2 bits de signe..)
Seuls 2N-1 bits sont nécessaires pour le résultat

Produit de 2 nombres en formats différents:

$Q_{K1} \times Q_{K2}$: Résultat en $\mathbf{Q_{K1+K2}}$

Si on souhaite un résultat en format Q_{K3} , on doit effectuer:

- **un décalage vers la droite de K1+K2-K3**
- **ou un décalage vers la gauche de N - (K1+K2-K3)**