الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي
Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

Architecture Matérielle de Commande des Machines

EE812 CM **Salle 107**

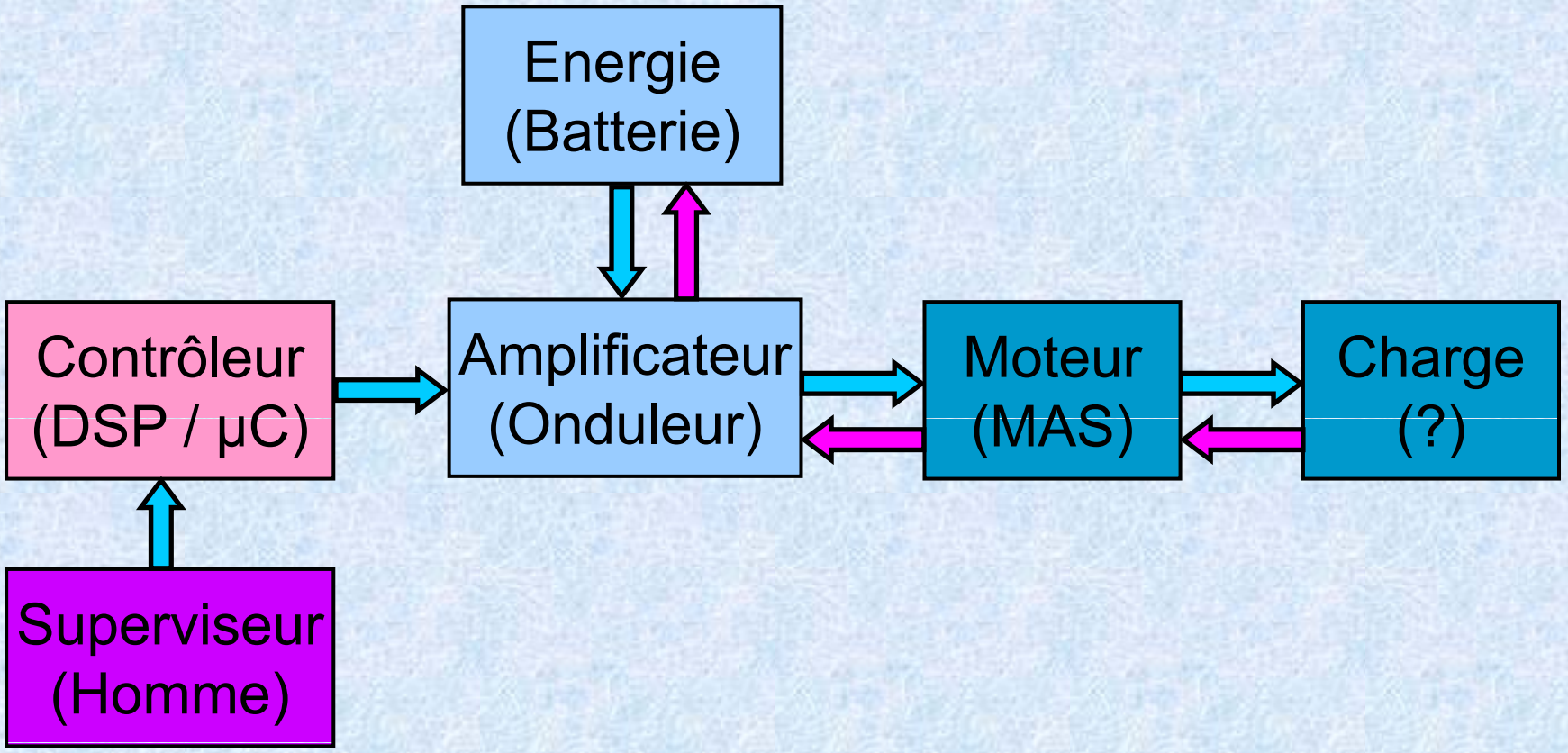
EE812 TP **Lab. de microprocesseurs**

Support de cours

http://baghli.com/doc_archi_cmde.php

Chaîne de conversion électromécanique

Schéma synoptique



Plan

- Généralités sur la commande de machines
  - Eléments d'une chaîne de conversion électromécanique
  - Motorisations électriques
  - Electronique de puissance
  - Commandes
  - Contrôle vectoriel de la MS

Plan

- Composants matériels :
  - DSP, µC, onduleur, hacheur
  - Capteurs : position, vitesse, accéléromètres et gyroscopes MEMS, courant, tension, induction)
  - Périphériques internes au dspic : ADC, PWM, I/O, UART…

Plan

- Composants logiciels :
  - Programmes de contrôle
  - Boucles de régulation
  - ISR

Plan

- Etude du système par l'exemple :
  - carte comportant un µC dspic 16 bits 30F3010, un onduleur triphasé, en circuit intégré (L6234), une interface UART-RS232 MAX3233 et des composants annexes :
  - http://baghli.com/dspic_archi_cmde.php

Carte d'essais

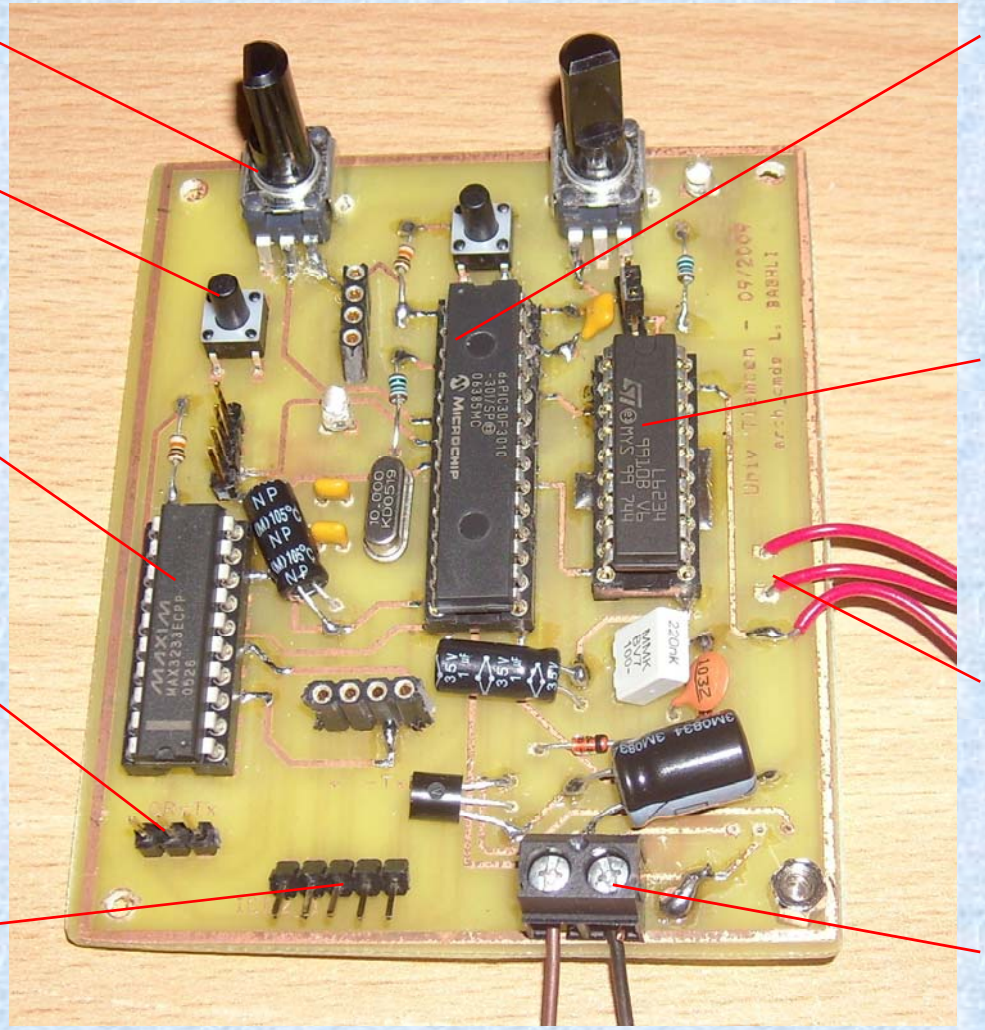
Potentiomètre

Bouton poussoir

Interface UART – RS232 (MAX3233)

Connecteur RS232 (vers PC)

Connecteur du programmateur PicKit2



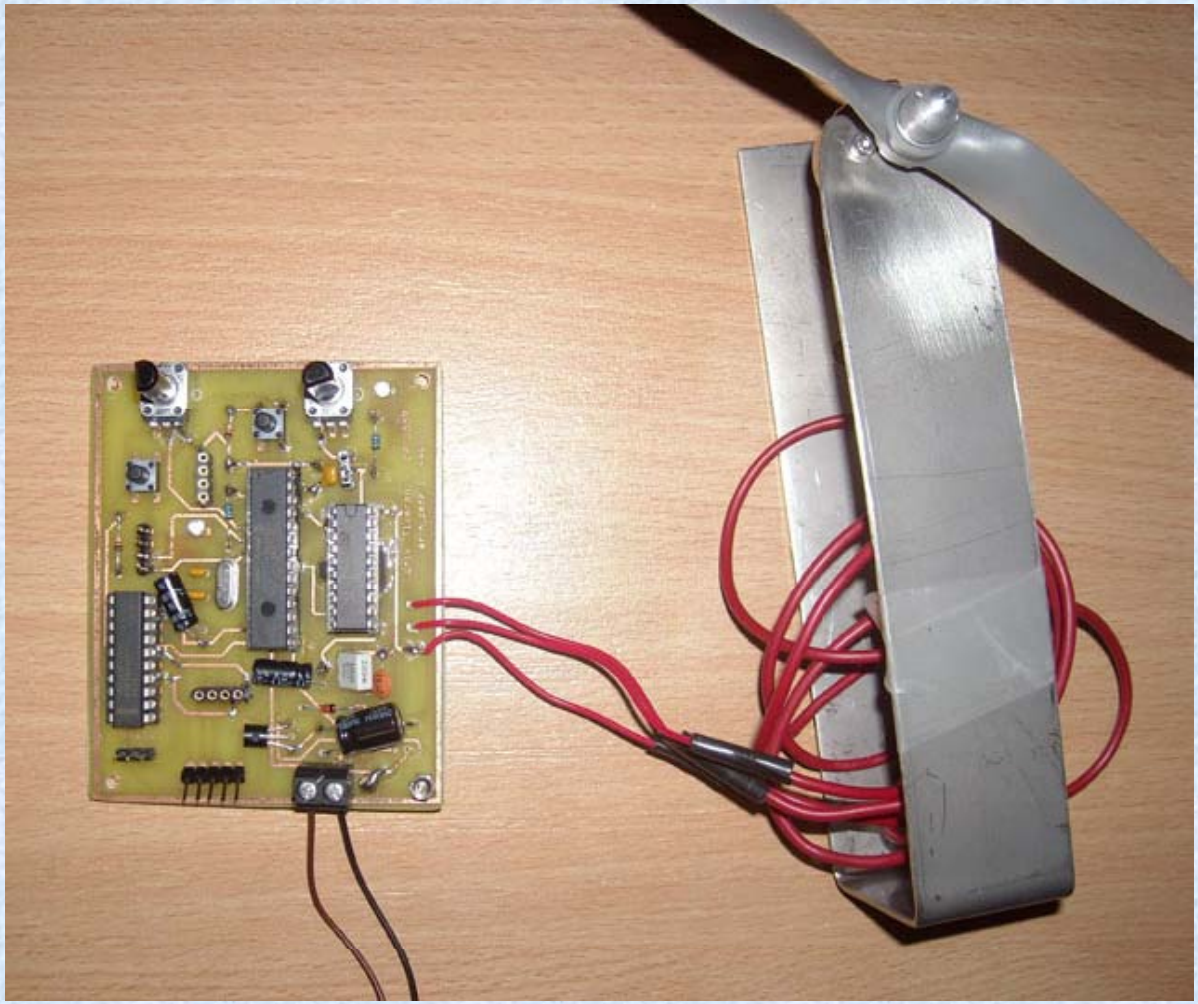
dspic 30F3010

Onduleur triphasé L6234

Branchement du moteur triphasé (sortie)

Alimentation 5V à 9 V (entrée)

Pilotage d’un MS BLDC



Plan

- Prise en main de l'outil de développement MPLAB / Microchip
  - Génération du projet, compilation, chargement, débogage pas à pas
  - http://www.microchip.com
- Architecture du dspic 30F3010 et programmation en C du µC dspic

Plan

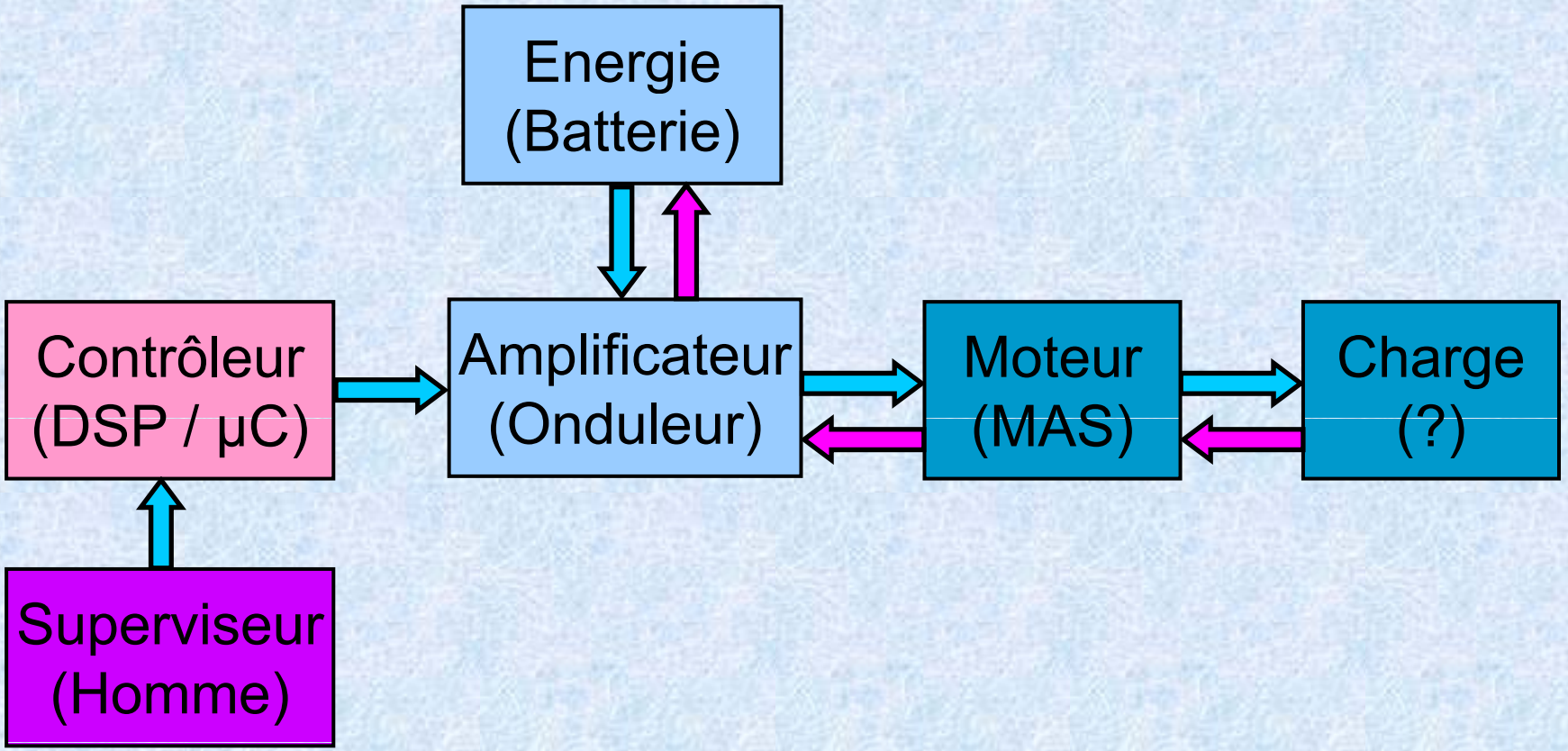
- Etude des différents périphériques du dspic :
  - Timers
  - Interruptions temps réel, ISR
  - Convertisseur analogique-numérique (CAN / ADC)
  - Liaison série asynchrone µC - UART / RS232 – PC
  - Modulation de largeur d'impulsion (MLI / PWM) pour la commande d'onduleur et de hacheur
  - Mise en œuvre de la communication avec le PC pour le pilotage à distance et pour la récupération des données en temps réel et l'affichage de courbes
  - Exemple d'une commande en V/f et création d'un champ tournant triphasé.

Généralités sur la commande de machines

- Eléments d'une chaîne de conversion électromécanique
- Motorisations électriques
- Electronique de puissance
- Commandes
- Contrôle vectoriel de la MS

Chaîne de conversion électromécanique

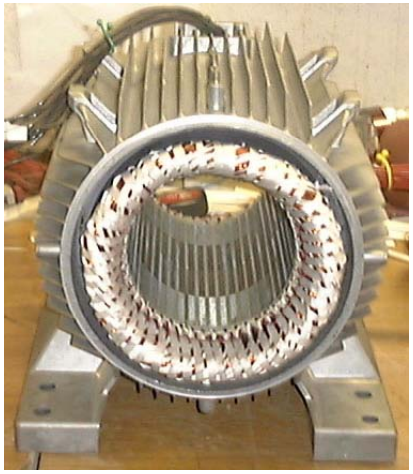
Schéma synoptique

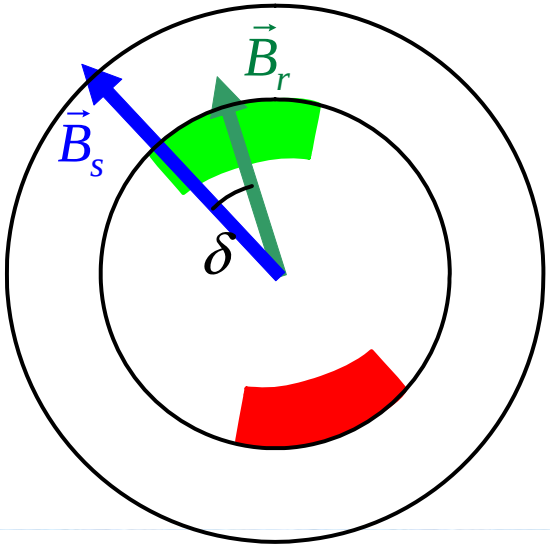


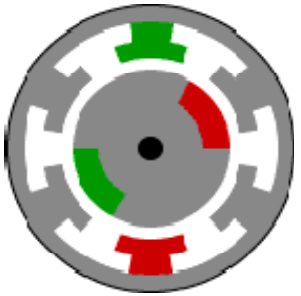
Moteurs

Constitution et principe de fonctionnement

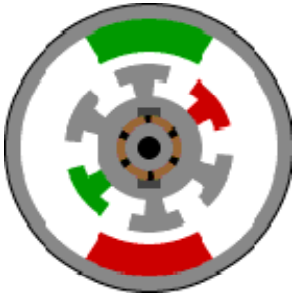
- MS
  - Stator triphasé, champ tournant
  - Rotor à excitation ext. ou à AP







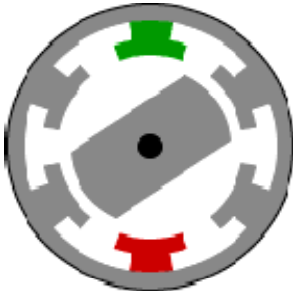
Comparaison des motorisations



MCC

MS

MAS

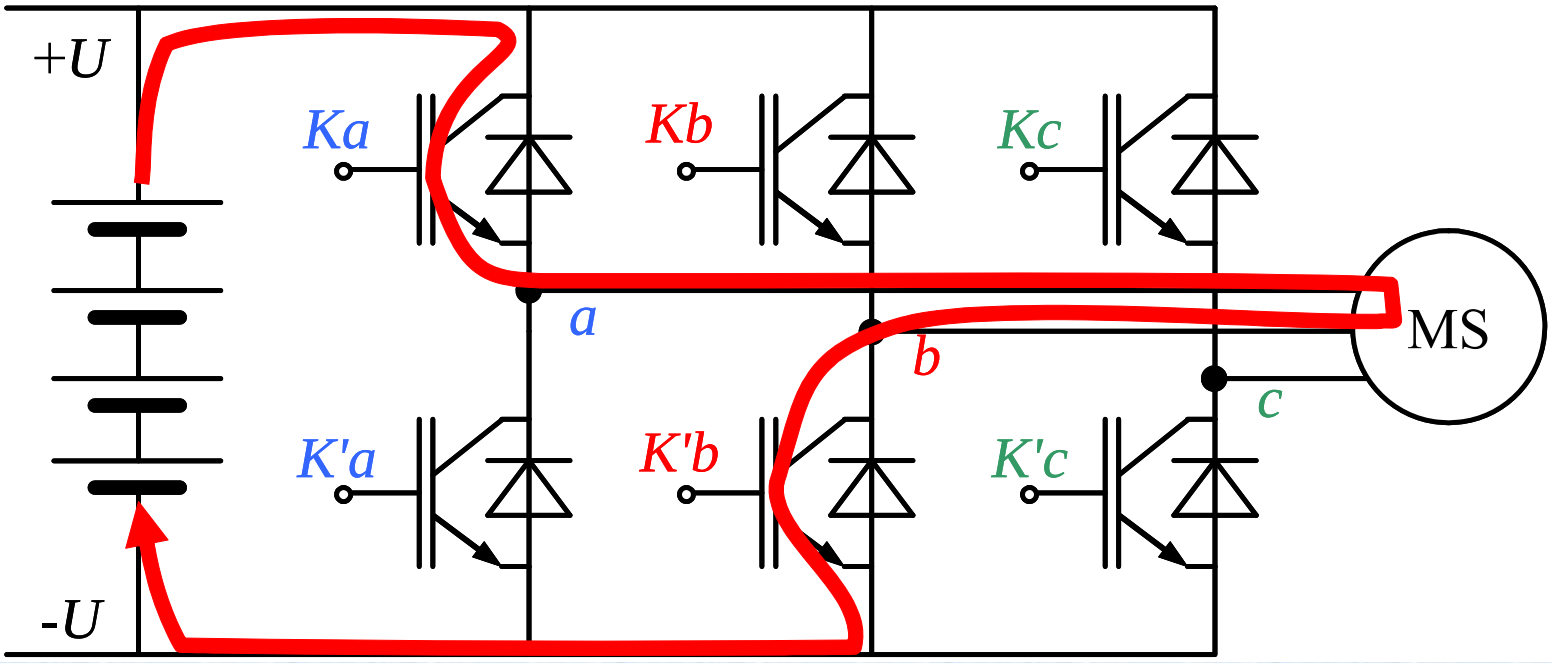


MRV

Type de machine	Simplicité de la Commande	Electronique de Puissance	Robustesse	Coût	Puissance massique
MCC	+	+	-	+	-
MS	-	0	-	+	+
MAS	-	0	+	-	-
MRV	-	-	+	-	-

Electronique de puissance

- Onduleur à IGBT ou MOSFET
- Tension du bus continu
- Commande BLDC ou Sinus



Commande de moteurs

- MCC
  - Simple, système découplé
  - Contrôle analogique ou numérique d'un hacheur
- MRV
  - Alimentation par phase
  - Connaître la position du rotor
- MS, MAS
  - Commande V/f (en industrie) mais pas en traction
  - Commande vectorielle ⇩ codeur incrémental / absolu / résolveur
  - Autopilotage (MS) ⇩ sondes à effet Hall

Commande vectorielle de MS

- MS de f.e.m. sinusoïdales
- Position connue avec précision
- Calcul lourd, transformation de Park
- $C_e = \mathrm{p}\left(\left(L_d - L_q\right) I_{ds} I_{qs} + \Psi_f I_{qs}\right)$
- Régulation du courant par hystérésis ou MLI

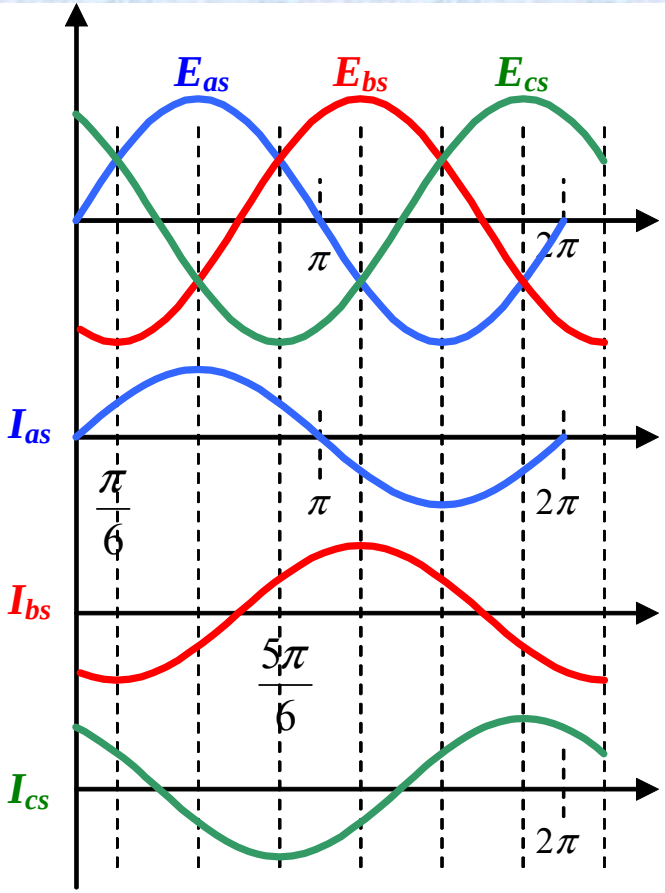
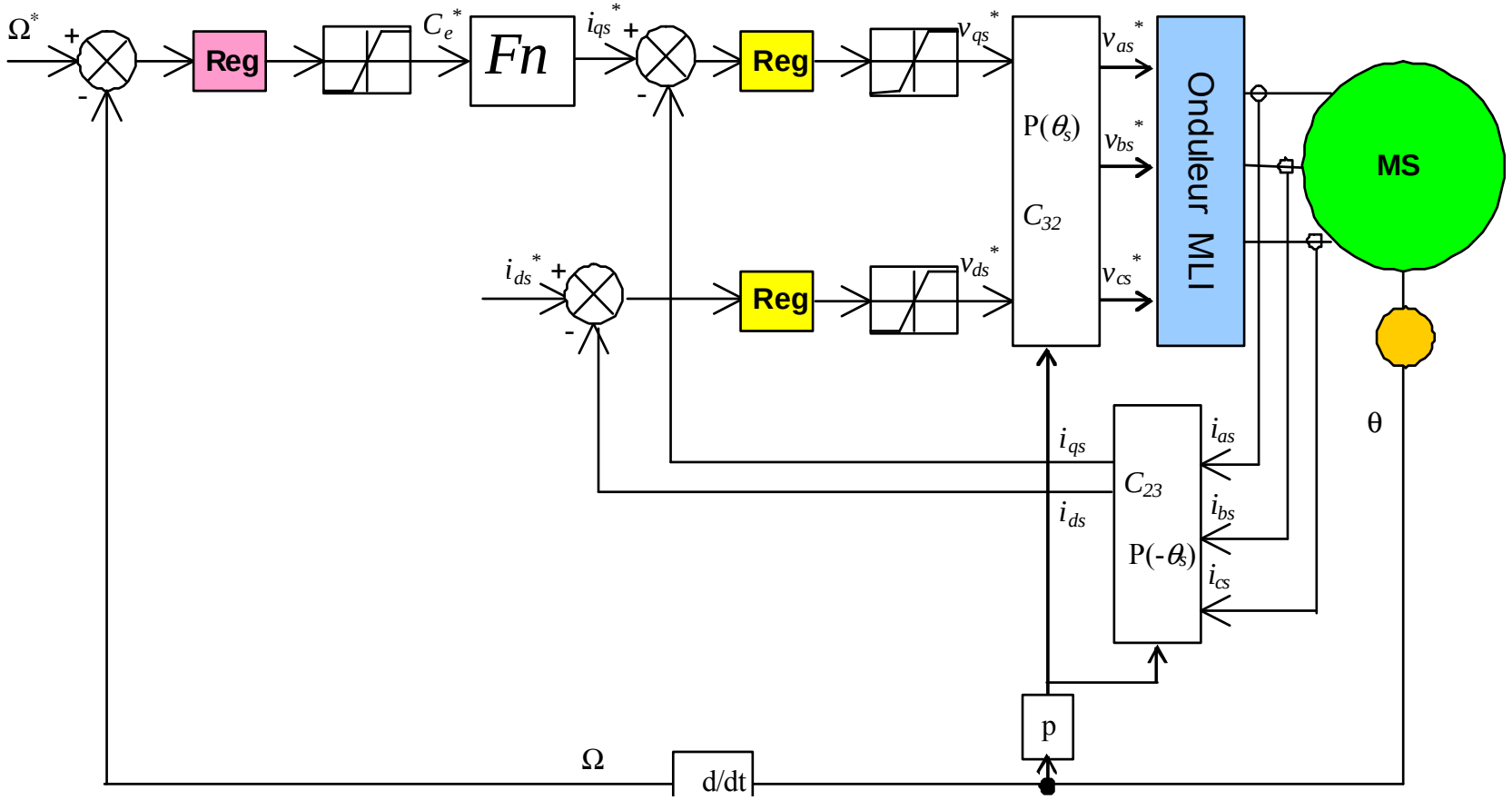
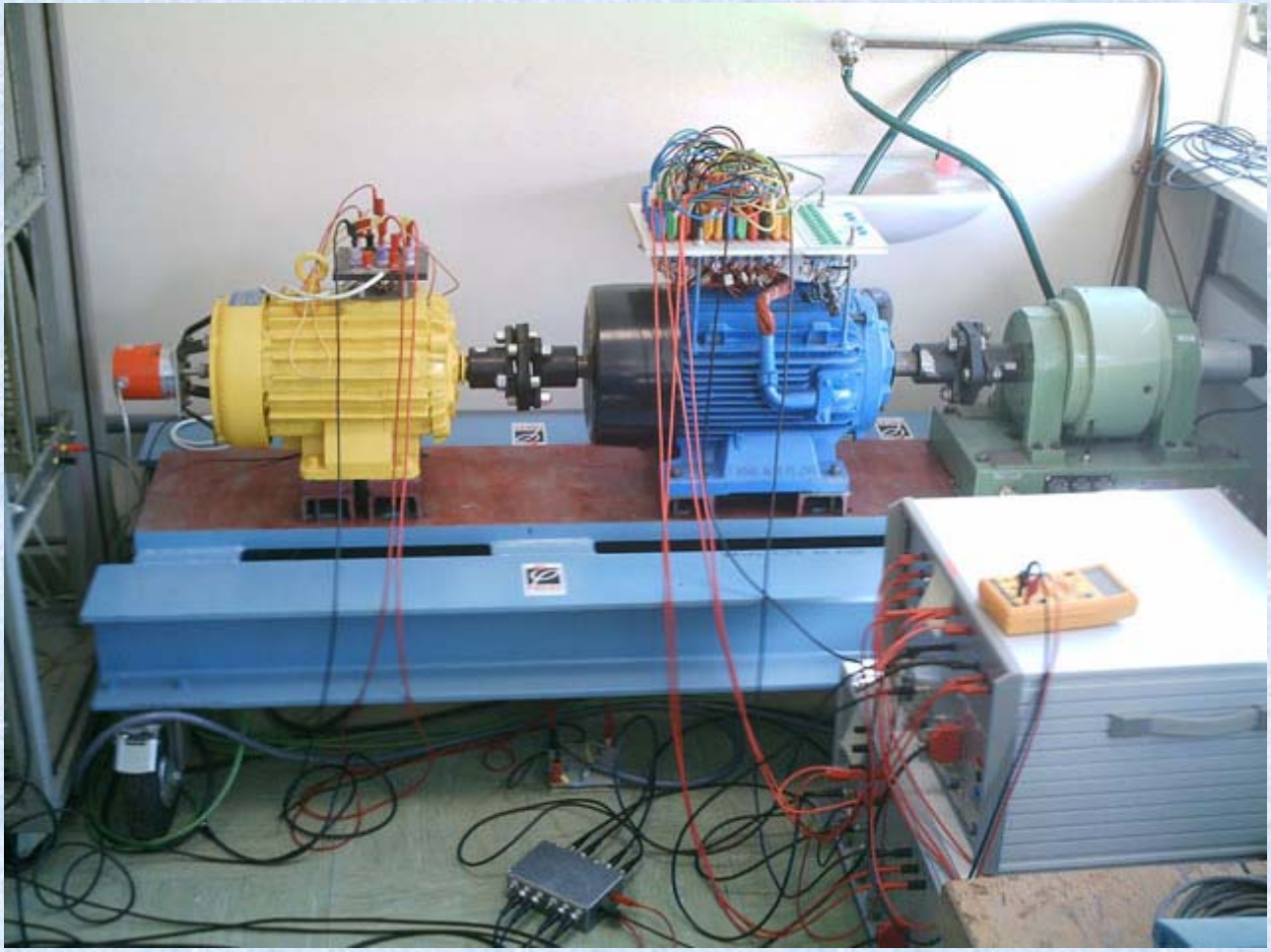


Schéma bloc d'un control



Chaîne de conversion électromécanique

Banc machines



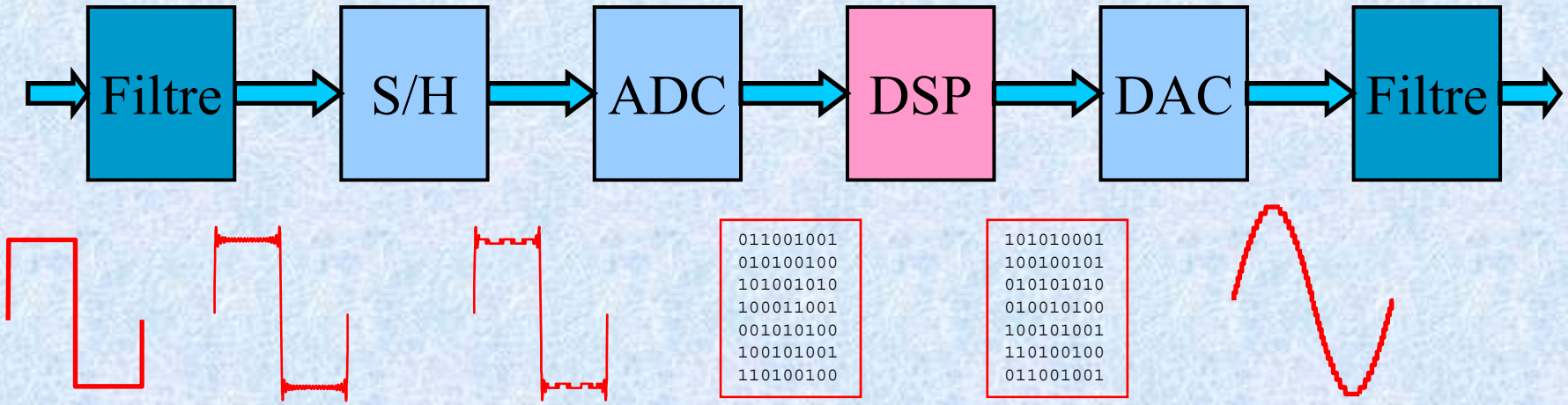
Composants matériels

- DSP, µC, onduleur, hacheur
- Capteurs : position, vitesse, accéléromètres et gyroscopes MEMS, courant, tension, induction)
- Périphériques internes au dspic : ADC, PWM, I/O, UART…

DSP : Introduction, Définitions

- **Digital Signal Processor**
  - Différence % au µP, µC
  - 4 à 400 USD
  - Taille $\rightarrow$2 cm^2 Silicium
  - CPU + CALU + périphériques intégrés
  - PWM, Watchdog, ADC, Timers, DMA, SCI, SPI, RAM, FLASH, IRDA, WiFi…

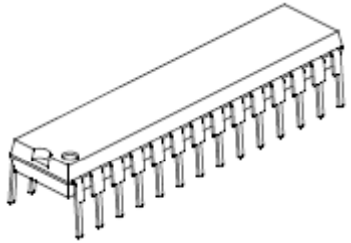
DSP : Introduction, Définitions



DSP et µC : applications

- **Communication**
  - Codage, suppression d'écho, fax, X25, téléphonie, PBX, VoIP, mobile (économie d'énergie)
- **Médecine**
  - Traitement d'imagerie RMN, Echographie (2D, 3D), Doppler…
- **Militaire**
  - Radar, guidage missiles.
  - Communications sécurisées, chiffrage, PGP…
- **Jeux**
  - Cartes graphiques, Animation 3D, textures (mapping), rotations, transparences…
- **Industrie**
  - Commande de moteur, robots, automates…

µC dsPIC 30F3010

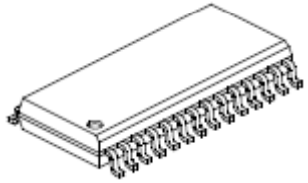


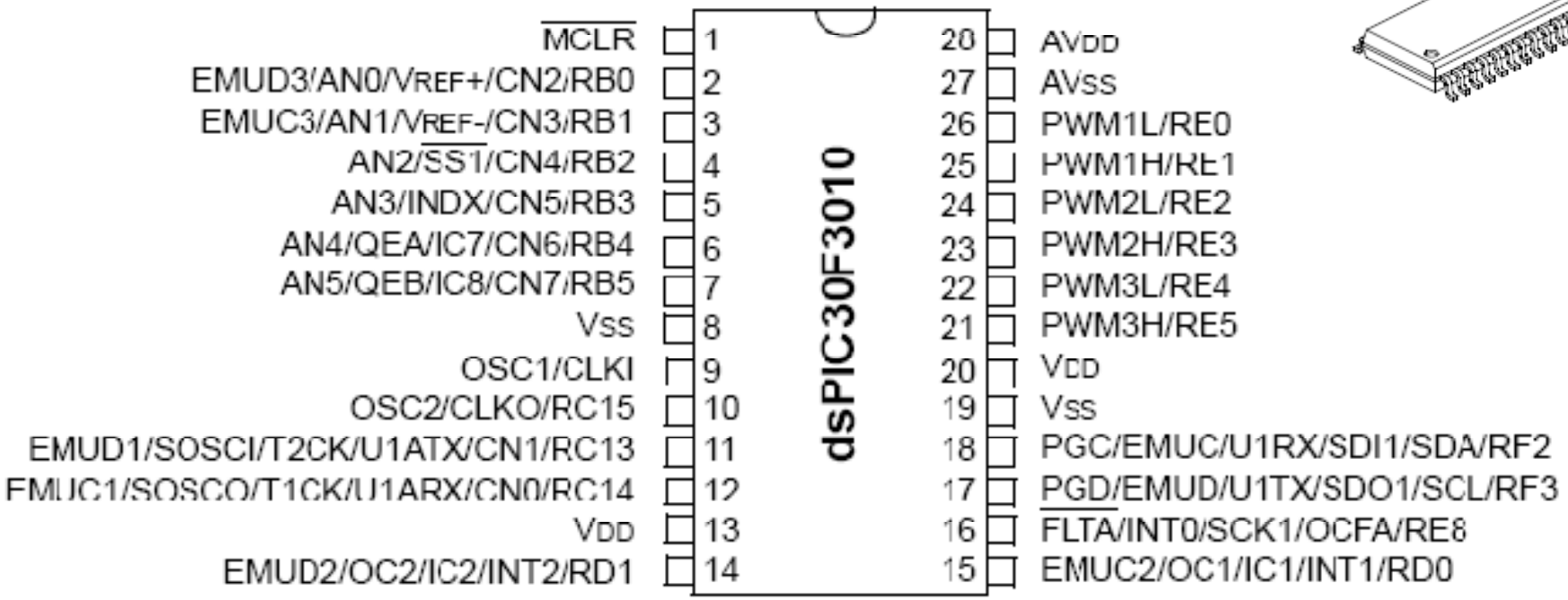
- **Document s**
  - dsPIC30F3010 DataSheet **70141c.pdf**
  - dsPIC30F Family Overview **70043F.pdf**
  - dsPIC30F Family Reference Manual **70046E.pdf**

70141c

70043F

70046E

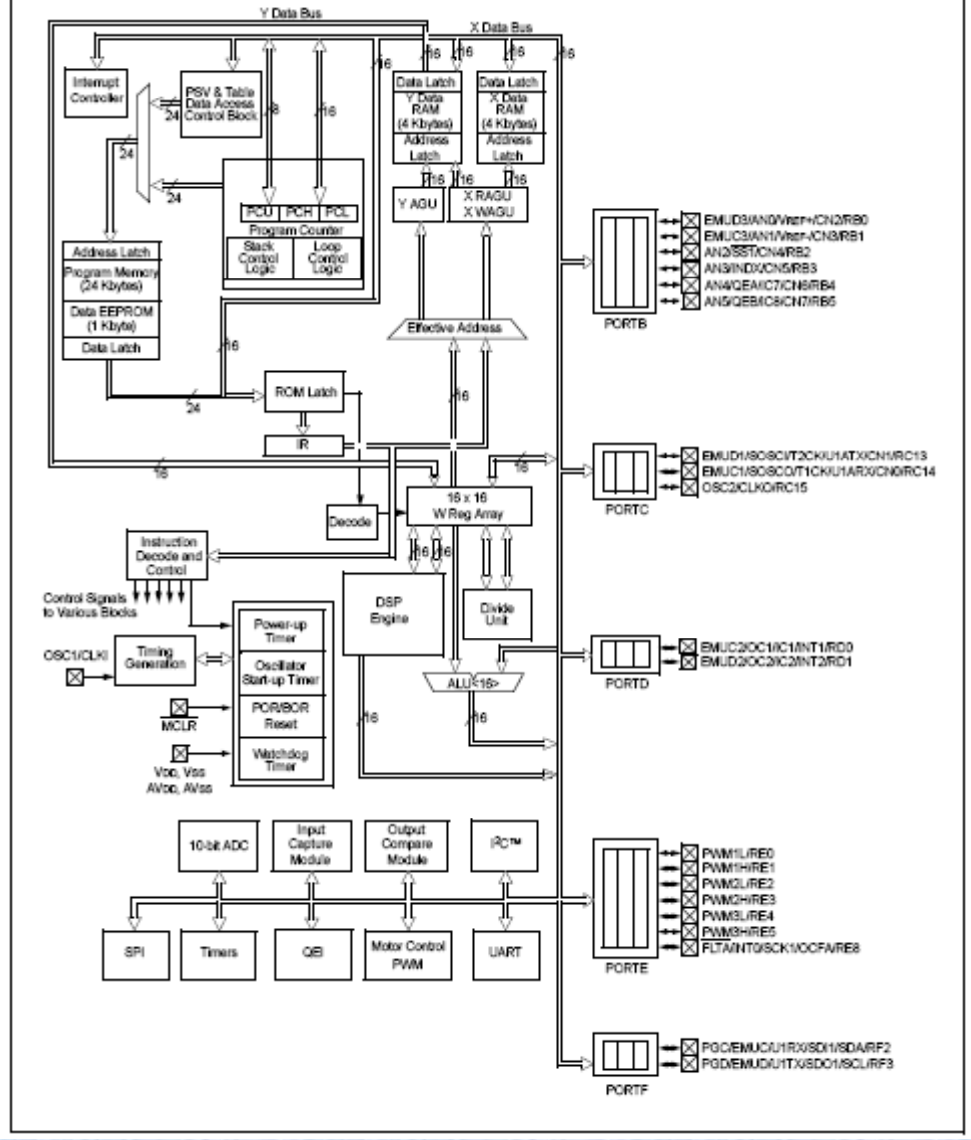




µC dsPIC 30F3010

- **Schéma bloc**
  - Page 11

FIGURE 1-2: dsPIC30F3010 BLOCK DIAGRAM



µC dsPIC 30F3010 – ports d'E/S

- **Schéma bloc**

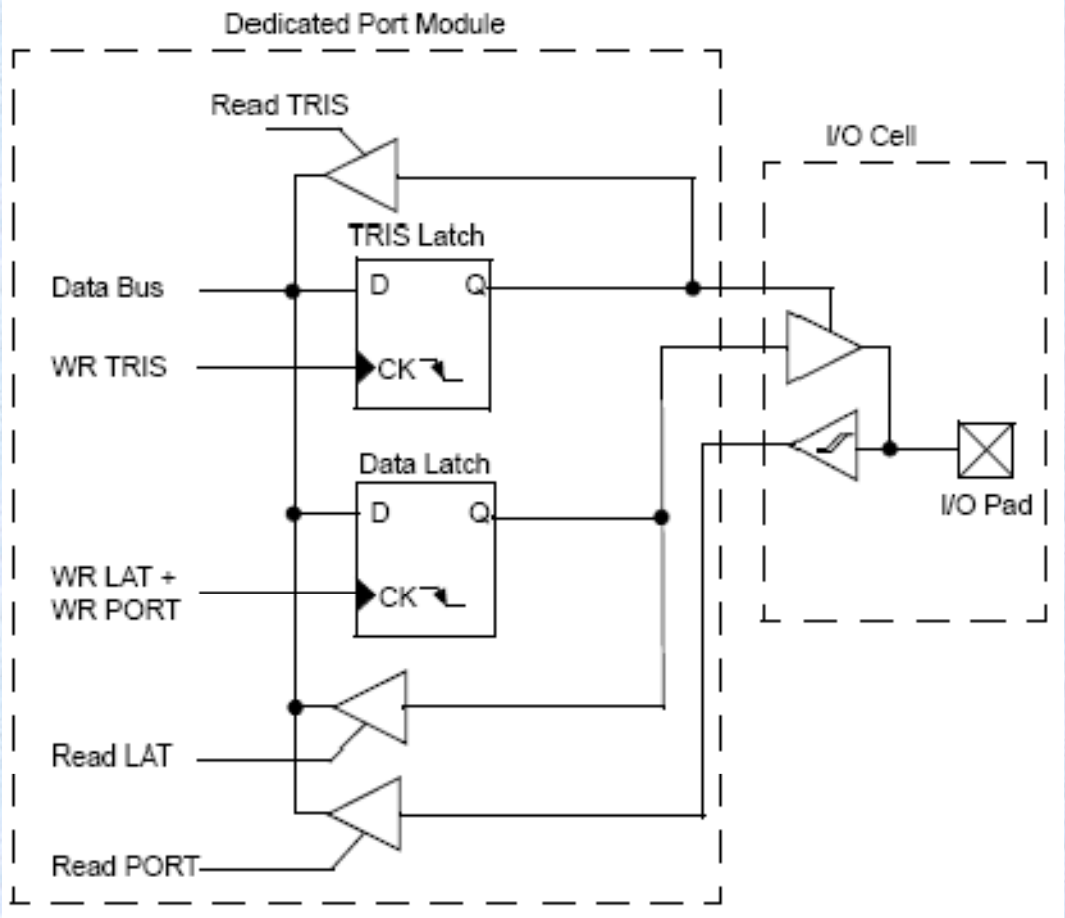
Ce dsPIC comporte plusieurs ports :

Port **B**, **C**, **D**, **E**, et **F**

Pour chacun, 3 registres sont associés :

TRIS**x**, PORT**x**, LAT**x**

ADPCFG : choix des pins en E/S logique ou en analogique



µC dsPIC 30F3010 - Timer

- **Schéma bloc**
  - Page 68
- **Registres**
  - Page 70

FIGURE 9-1: 16-BIT TIMER1 MODULE BLOCK DIAGRAM (TYPE A TIMER)

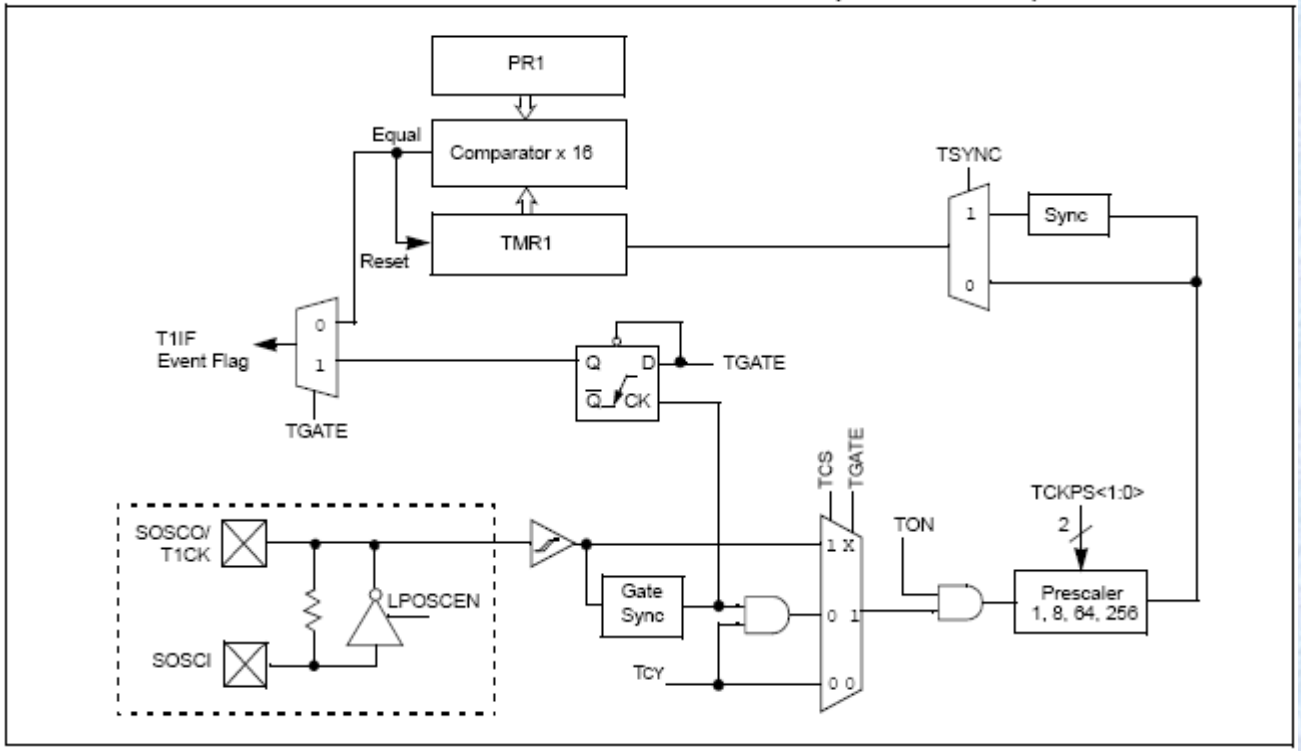


TABLE 9-1: TIMER1 REGISTER MAP(1)

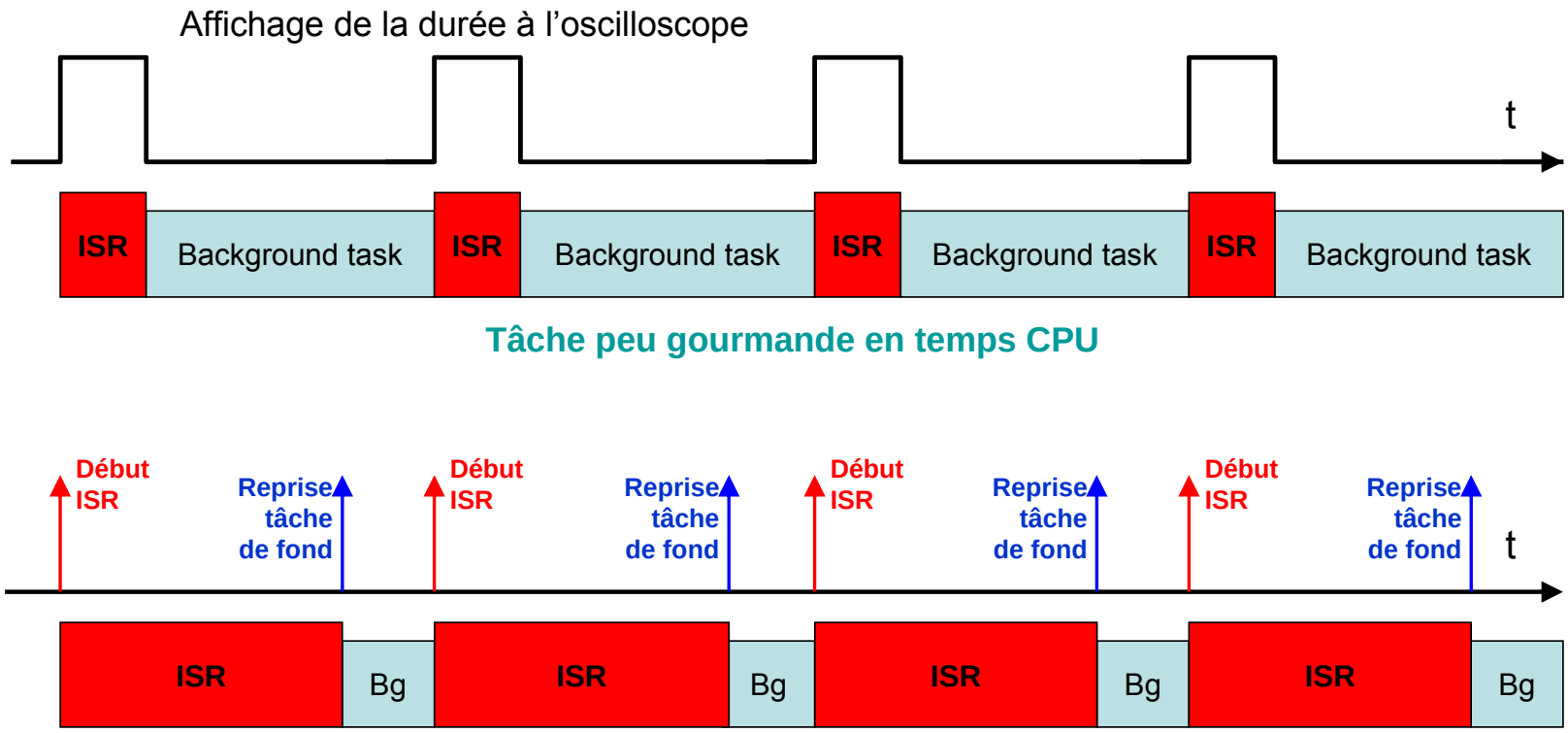
SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TMR1	0100	Timer1 Register															
PR1	0102	Period Register 1															
T1CON	0104	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—

Legend: u = uninitialized bit; — = unimplemented bit, read as '0'

Note 1: Refer to "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

µC dsPIC 30F3010 – Interruption

- **ISR Interrupt Service Routine**



Tâche peu gourmande en temps CPU

µC dsPIC 30F3010 – Interruption

- **Registres** IF, IEC, IPC (page 48)

TABLE 5-2: INTERRUPT CONTROLLER REGISTER MAP(1)

SFR Name	ADR	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—
INTCON2	0082	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	—	—	INT2EP	INT1EP	INT0EP
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF
IFS1	0086	—	—	—	—	—	—	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF
IFS2	0088	—	—	—	—	FLTAIF	—	—	QEIIF	PWMIF	—	—	—	—	—	—	—
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE
IEC1	008E	—	—	—	—	—	—	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE
IEC2	0090	—	—	—	—	FLTAIE	—	—	QEIIE	PWMIE	—	—	—	—	—	—	—
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>		
IPC1	0096	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>		
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>		
IPC3	009A	—	CNIP<2:0>			—	MI2CIP<2:0>			—	SI2CIP<2:0>			—	NVMIP<2:0>		
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>		
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			—	OC4IP<2:0>		
IPC6	00A0	—	—	—	—	—	—	—	—	—	U2TXIP<2:0>			—	U2RXIP<2:0>		
IPC7	00A2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
IPC8	00A4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
IPC9	00A6	—	PWMIP<2:0>			—	—	—	—	—	INT41IP<2:0>			—	INT3IP<2:0>		
IPC10	00A8	—	FLTAIP<2:0>			—	—	—	—	—	—	—	—	—	QEIIP<2:0>		
IPC11	00AA	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Legend: — = unimplemented, read as '0'

Note 1: Refer to "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

Exemple d'ISR / Timer

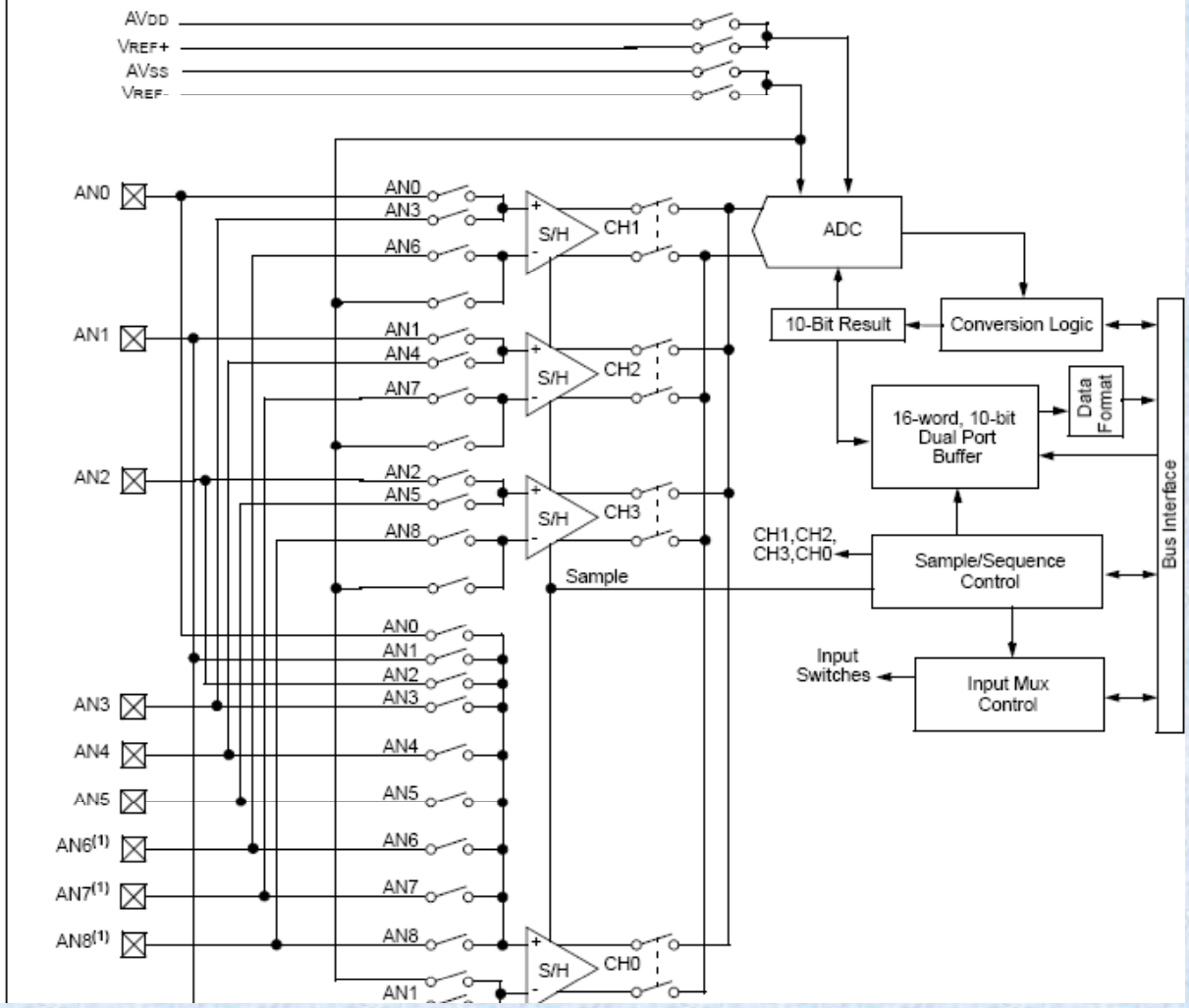
```
//-----------------------------------------------------------------------
void initTimer(void)
{
// Timer1 pour l ISR des 200 us
          T1CON = 0;                    // ensure Timer 1 is in reset state, internal
timer clock Fosc/4, no prescale
          TMR1  = 0;                                                  // RAZ Timer1
          _T1IF = 0;          // reset Timer 1 interrupt flag
          _T1IP = 4;          // set Timer1 interrupt priority level to 4
          _T1IE = 1;          // enable Timer 1 interrupt
          PR1 = T1Period;                          // set Timer 1 period register
          T1CONbits.TON = 1;  // enable Timer 1 and start the count
}

//-----------------------------------------------------------------------
// Timer1 interrupt, ISR toutes les 200 us
//-----------------------------------------------------------------------
void __attribute__((interrupt, auto_psv)) _T1Interrupt( void )
{
          InfoLED = 1;
          _T1IF = 0;
// do some work
// ...
// ...
          InfoLED = 0;
}
```

µC dsPIC 30F3010 - ADC

- **Convertisseur Analogique – Numérique (CAN)**
  - Page 128
  - 10 bits, 4 S/H, 1 ADC
  - Immediate SOC
  - Synchronised SOC (PWM event, timer
  - Interrupt on EOC

FIGURE 19-1: 10-BIT HIGH-SPEED ADC FUNCTIONAL BLOCK DIAGRAM



µC dsPIC 30F3010 - ADC

Control Registers

The A/D module has six Control and Status registers. These registers are:

- ADCON1: A/D Control Register 1
- ADCON2: A/D Control Register 2
- ADCON3: A/D Control Register 3
- ADCHS: A/D Input Channel Select Register
- ADPCFG: A/D Port Configuration Register
- ADCSSL: A/D Input Scan Selection Register

The ADCON1, ADCON2 and ADCON3 registers control the operation of the A/D module. The ADCHS register selects the input pins to be connected to the S/H amplifiers. The ADPCFG register configures the analog input pins as analog inputs or as digital I/O. The ADCSSL register selects inputs to be sequentially scanned.

A/D Result Buffer

The module contains a 16-word dual port RAM, called ADCBUF, to buffer the A/D results. The 16 buffer locations are referred to as ADCBUF0, ADCBUF1, ADCBUF2, ..., ADCBUFE, ADCBUFF.

µC dsPIC 30F3010 - ADC

- **Registres** (page 137)

TABLE 19-2: ADC REGISTER MAP(1)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCBUF0	0280	—	—	—	—	—	—	ADC Data Buffer 0									
ADCBUF1	0282	—	—	—	—	—	—	ADC Data Buffer 1									
ADCBUF2	0284	—	—	—	—	—	—	ADC Data Buffer 2									
ADCBUF3	0286	—	—	—	—	—	—	ADC Data Buffer 3									
ADCBUF4	0288	—	—	—	—	—	—	ADC Data Buffer 4									
ADCBUF5	028A	—	—	—	—	—	—	ADC Data Buffer 5									
ADCBUF6	028C	—	—	—	—	—	—	ADC Data Buffer 6									
ADCBUF7	028E	—	—	—	—	—	—	ADC Data Buffer 7									
ADCBUF8	0290	—	—	—	—	—	—	ADC Data Buffer 8									
ADCBUF9	0292	—	—	—	—	—	—	ADC Data Buffer 9									
ADCBUFA	0294	—	—	—	—	—	—	ADC Data Buffer 10									
ADCBUFB	0296	—	—	—	—	—	—	ADC Data Buffer 11									
ADCBUFC	0298	—	—	—	—	—	—	ADC Data Buffer 12									
ADCBUFD	029A	—	—	—	—	—	—	ADC Data Buffer 13									
ADCBUFE	029C	—	—	—	—	—	—	ADC Data Buffer 14									
ADCBUFF	029E	—	—	—	—	—	—	ADC Data Buffer 15									
ADCON1	02A0	ADON	—	ADSIDL	—	—	—	FORM<1:0>		SSRC<2:0>			—	SIMSAM	ASAM	SAMP	DONE
ADCON2	02A2	VCFG<2:0>			—	—	CSCNA	CHPS<1:0>		BUFS	—	SMPI<3:0>				BUFM	ALTS
ADCON3	02A4	—	—	—	SAMC<4:0>					ADRC	—	ADCS<5:0>					
ADCHS	02A6	CH123NB<1:0>		CH123SB	CH0NB	CH0SB<3:0>				CH123NA<1:0>		CH123SA	CH0NA	CH0SA<3:0>			
ADPCFG	02A8	—	—	—	—	—	—	—	PCFG8(2)	PCFG7(2)	PCFG6(2)	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
ADCSSL	02AA	—	—	—	—	—	—	—	CSSL8(2)	CSSL7(2)	CSSL6(2)	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0

Legend: u = uninitialized bit; — = unimplemented bit, read as '0'

Note 1: Refer to "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

2: These bits are not available on dsPIC30F3010 devices.

µC dsPIC 30F3010 - ADC

- **ADCON1**

bit 7-5 **SSRC<2:0>:** Conversion Trigger Source Select bits
111 = Internal counter ends sampling and starts conversion (auto convert)
110 = Reserved
101 = Reserved
100 = Reserved
011 = Motor Control PWM interval ends sampling and starts conversion
010 = GP Timer3 compare ends sampling and starts conversion
001 = Active transition on INT0 pin ends sampling and starts conversion
000 = Clearing SAMP bit ends sampling and starts conversion

Exemple d'ADC SOC 1/2

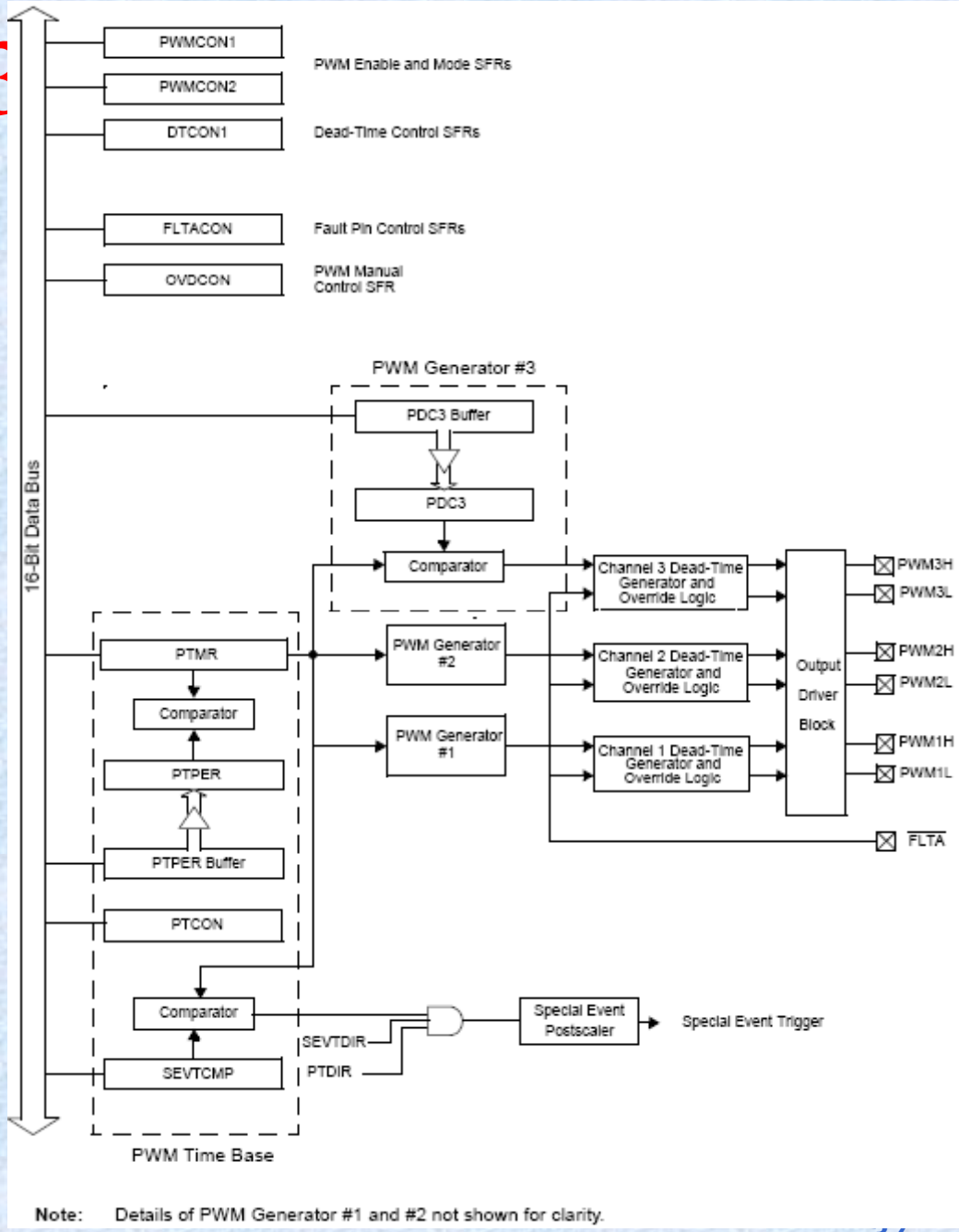
```
void InitADC10()
{
  // config ADC en lancement immédiat
  ADCON1 = 0x0004;    // 4 ch simultanés, ADC Off for configuring
                      // ASAM bit = 1 implies sampling starts immediately after last
                      // conversion is done
  ADCON2 = 0x0200;    // simulataneous sample 4 channels, ADC INTerrupt à chaque EOC=100 us
  ADCHS  = 0x0022;    // AN2/RB2 Ch0 Bref, AN3/RB3 Ch1 Bmes, AN4/RB4 Ch2 NC, AN5/RB5 Ch3 NC
  ADCON3 = 0x0080;    // Tad = internal RC (4uS)
  _ADON = 1;          // turn ADC ON
}
//-------------------------------------------------------------------------
//Main routine
int main()
{
  setup_ports();
  InitADC10();
  RunningLED=1;                          // LED run = On
  RunningLED=0;                          // LED run = Off
  while(1)
      {
      DelayNmSec(100);
      ADCON1bits.SAMP = 0;               // start conversion
      while(!ADCON1bits.DONE) {}         // attend la fin
      ValAdc=ADCBUF0;
      }   // end of while (1)
} // end of main
```

Exemple d'ADC SOC 2/2

```
void InitADC10()
{
  // config ADC en lancement immédiat
  ADCON1 = 0x0000;    // 4 ch simultanés, ADC Off for configuring
                      // ASAM bit = 0 implies SAMP=0 ends sampling and start converting
  ADCON2 = 0x0200;    // simulataneous sample 4 channels, ADC INTerrupt à chaque EOC=100 us
  ADCHS  = 0x0022;    // AN2/RB2 Ch0 Bref, AN3/RB3 Ch1 Bmes, AN4/RB4 Ch2 NC, AN5/RB5 Ch3 NC
  ADCON3 = 0x0080;    // Tad = internal RC (4uS)
  _ADON = 1;          // turn ADC ON
}
//-----------------------------------------------------------------------------
//Main routine
int main()
{
  setup_ports();
  InitADC10();
  RunningLED=1;                         // LED run = On
  RunningLED=0;                         // LED run = Off
  while(1)
      {
      ADCON1bits.SAMP = 1;              // start sampling
      DelayNmSec(100);
      ADCON1bits.SAMP = 0;              // start conversion
      while(!ADCON1bits.DONE) {}        // attend la fin
      ValAdc=ADCBUF0;
      }   // end of while (1)
} // end of main
```

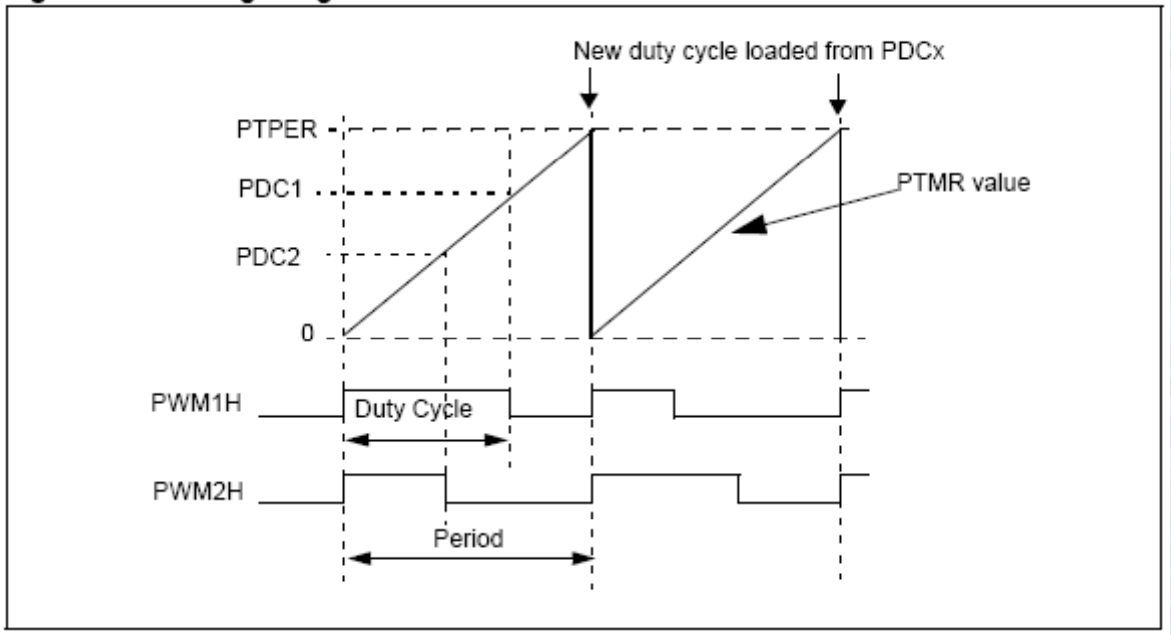
µC dsPIC 30F3

- **Motor Control PWM**
  - Page 95
  - Génère 6 signaux PWM synchronisés
  - 6 = 3 phases x (Lo et Hi)
  - Commande de MAS, MS, BLDC, MRV
  - PWM centrée ou non
  - Dead-time
  - Output Override
  - Evènements (SOC,…)



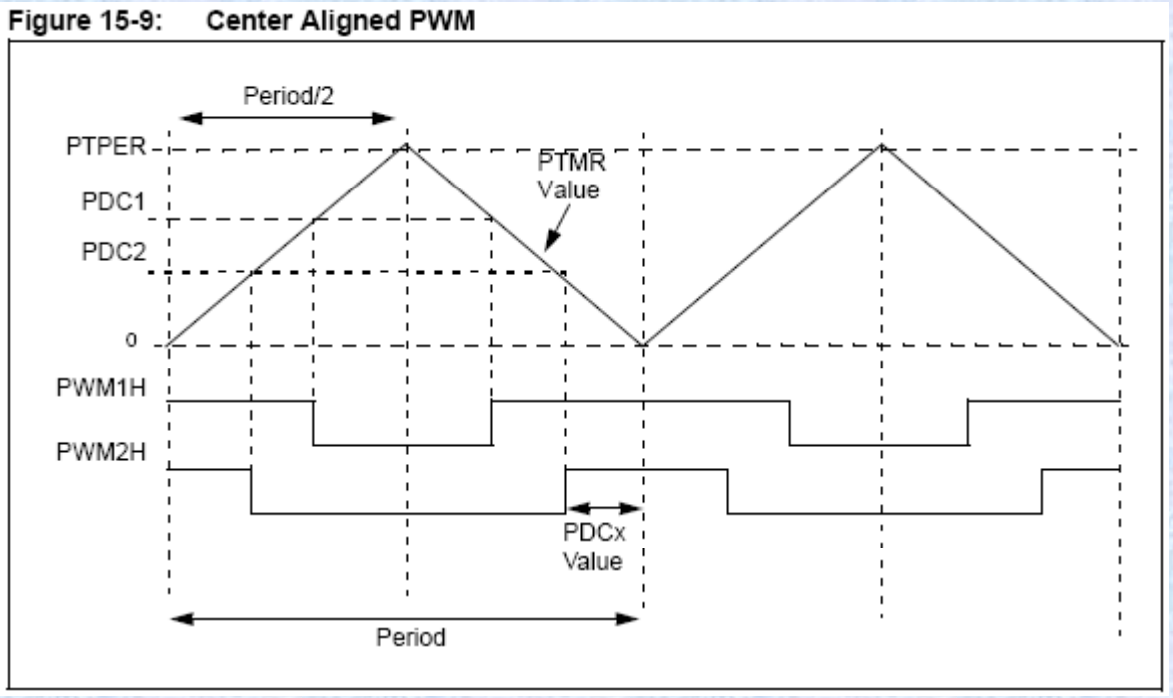
µC dsPIC 30F3010 - PWM

Figure 15-7: Edge-Aligned PWM



µC dsPIC 30F3010 - PWM

Figure 15-9: Center Aligned PWM



µC dsPIC 30F3010 - PWM

- **Registres** (page 105)

TABLE 15-1: PWM REGISTER MAP(1)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PTCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>	
PTMR	01C2	PTDIR	PWM Timer Count Value														
PTPER	01C4	—	PWM Time Base Period Register														
SEVTCMP	01C6	SEVTDIR	PWM Special Event Compare Register														
PWMCON1	01C8	—	—	—	—	—	PTMOD3	PTMOD2	PTMOD1	—	PEN3H	PEN2H	PEN1H	—	PEN3L	PEN2L	PEN1L
PWMCON2	01CA	—	—	—	—	SEVOPS<3:0>				—	—	—	—	—	—	OSYNC	UDIS
DTCON1	01CC	—	—	—	—	—	—	—	—	DTAPS<1:0>		Dead-Time A Value					
FLTACON	01D0	—	—	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	—	FAEN3	FAEN2	FAEN1
OVDCON	01D4	—	—	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	—	—	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L
PDC1	01D6	PWM Duty Cycle 1 Register															
PDC2	01D8	PWM Duty Cycle 2 Register															
PDC3	01DA	PWM Duty Cycle 3 Register															

Legend: — = unimplemented bit, read as '0'

Note 1: Refer to "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

Exemple de PWM

```
void InitMCPWM()
{
  PTPER = HalfDUTY;   // set the pwm period register, ne pas oublier la double précision
//    OVDCON = 0xFFFF;    // Cmde MLI, no effect of OVDCON
  OVDCON = 0x00FF;    // allow control using OVD register (active low : PWM pins ie 0)
//    PWMCON1 = 0x0700;       // disable PWMs
  PWMCON1= 0x0770;    // enable PWM outputs (mais pas les LOW)
  PDC1=HalfDUTY; PDC2=HalfDUTY; PDC3=HalfDUTY;    // init rapport cyclique 50%
  EnableL6234=0;                          // disable L298
  PWMCON2 = 0x0000;       // 1 PWM values
  PTCON = 0x8002;         // start PWM symetrique
}
//-----------------------------------------------------------------------------

  OVDCON = 0xFFFF;    // Cmde MLI, no effect of OVDCON

  OVDCON = 0x00FF;    // allow control using OVD register (active low : PWM pins ie 0)
```

Exemple d'ADC SOC + PWM 1/3

```
//-----------------------------------------------------------------------------
//    Setup the ADC registers for :
//    4 channels conversion
//    PWM trigger starts conversion
//    AD interrupt is set and buffer is read in the interrupt
//-----------------------------------------------------------------------------
void InitADC10()
{
  ADCON1 = 0x006F;    // PWM starts conversion, 4 ch simultanés, ADC Off for configuring
  ADCON2 = 0x0200;    // simulataneous sample 4 channels, ADC INTerrupt à chaque EOC=100 us
  ADCHS   = 0x0022;   // AN2/RB2 Ch0 Bref, AN3/RB3 Ch1 Bmes, AN4/RB4 Ch2 NC, AN5/RB5 Ch3 NC
  ADCON3 = 0x0080;    // Tad = internal RC (4uS)
  _ADIF = 0;          // Adc int flag Off
  _ADIE = 1;          // Adc int On
  _ADON = 1;          // turn ADC ON
}
```

Exemple d'ADC SOC + PWM 2/3

```
//-----------------------------------------------------------------------------
//  InitMCPWM, intializes the PWM as follows:
//  FPWM = 16 khz voir en haut
//  Independant PWMs
//  Set ADC to be triggered by PWM special trigger
//-----------------------------------------------------------------------------
void InitMCPWM()
{
  PTPER = HalfDUTY;   // set the pwm period register, ne pas oublier la double précision
  OVDCON = 0x00FF;    // allow control using OVD register (active low pour PWM pins ie 0)
//    PWMCON1 = 0x0700;       // disable PWMs
  PWMCON1= 0x0770;            // enable PWM outputs (mais pas les LOW)
  PDC1=HalfDUTY; PDC2=HalfDUTY; PDC3=HalfDUTY;                    // init sans rien, apres
une regul ça change
  EnableL6234=0;                                          // disable L298
  SEVTCMP = PTPER;            // set ADC to trigeer at ...
  PWMCON2 = 0x0000;           // 1 PWM values
  PTCON = 0x8002;             // start PWM symetrique
}
```

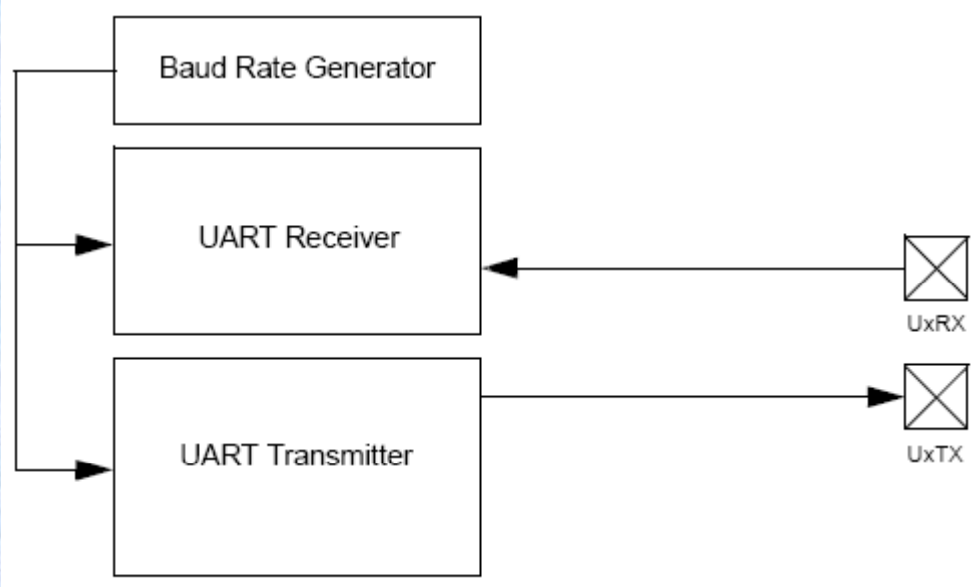
Exemple d'ADC SOC + PWM 3/3

```
//-----------------------------------------------------------------------
//    The ADC interrupt reads the demand pot value.
// tt est synchrone % à cette int de ADC int =2*PWMperiod=2*62.5 us=125 us
//-----------------------------------------------------------------------
void __attribute__((interrupt, auto_psv)) _ADCInterrupt ()
{
  InfoLED=1;
  _ADIF = 0;
  k_V_f=ADCBUF0<<2;                    // 4.12 pu 4096=1.0=6V
  fs=ADCBUF1<<2;                       // 4.12 pu 4096=1.0=20 Hz
  InfoLED=0;
}
//-------------------------------------------------------------------------
//Main routine
int main()
{
  setup_ports();
  InitADC10();
  InitMCPWM();
  RunningLED=1;                                          // LED run = On
  RunningLED=0;                                          // LED run = Off
  while(1)
      {
      }   // end of while (1)
} // end of main
```

µC dsPIC 30F3010 - UART

Description

L'UART est un système asynchrone full-duplex qui peut communiquer avec des périphériques, comme des PC, des interfaces RS-232 et RS-485.



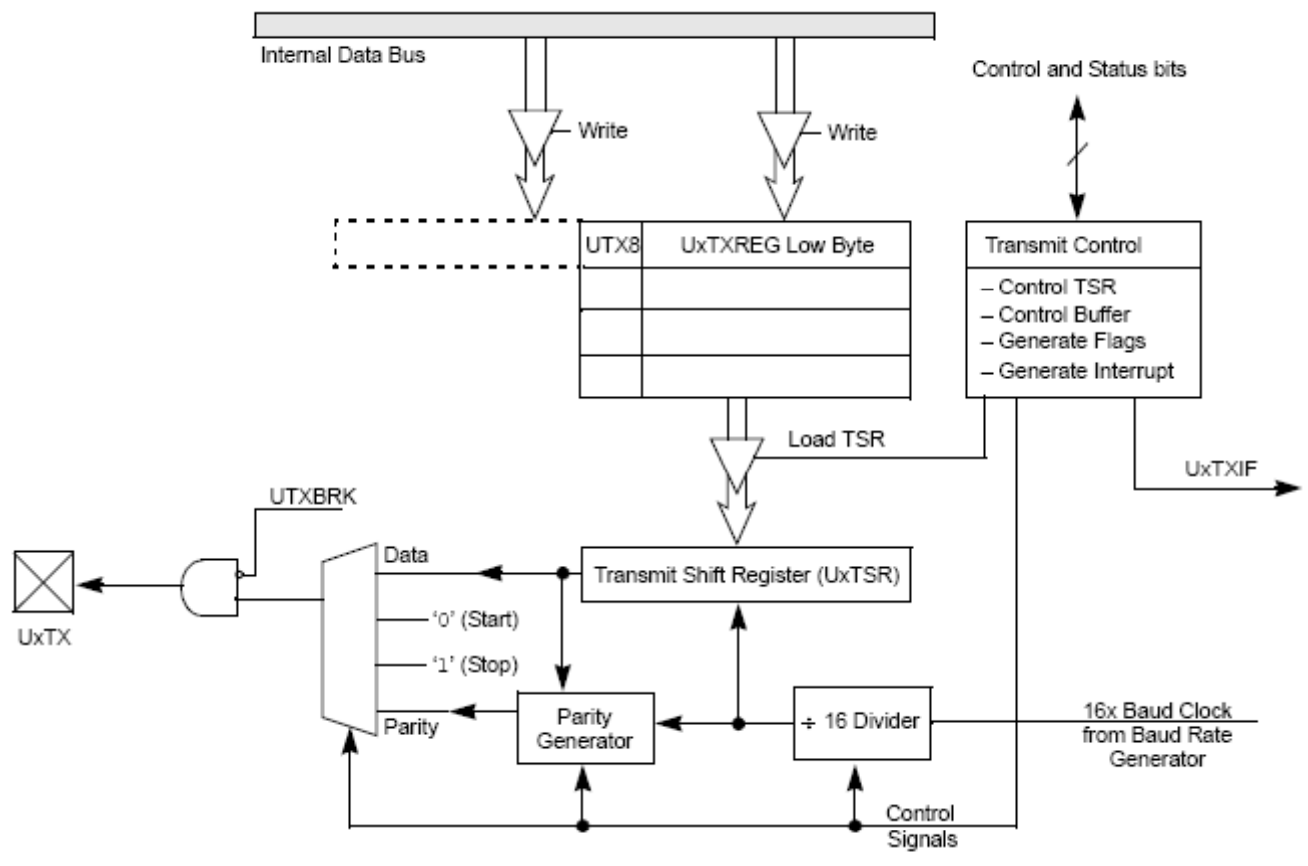
µC dsPIC 30F3010 - UART

Features (caractéristiques)

- Full-duplex 8- or 9-bit data transmission through the UxTX and UxRX pins
- Even, Odd or No Parity options (for 8-bit data)
- One or two Stop bits
- Fully integrated Baud Rate Generator with 16-bit prescaler
- Baud rates ranging from 29 bps to 1.875 Mbps at F_{CY} = 30 MHz
- 4-deep First-In-First-Out (FIFO) transmit data buffer
- 4-deep FIFO receive data buffer
- Parity, Framing and Buffer Overrun error detection
- Support for 9-bit mode with Address Detect (9th bit = 1)
- Transmit and Receive Interrupts
- Loopback mode for diagnostic support

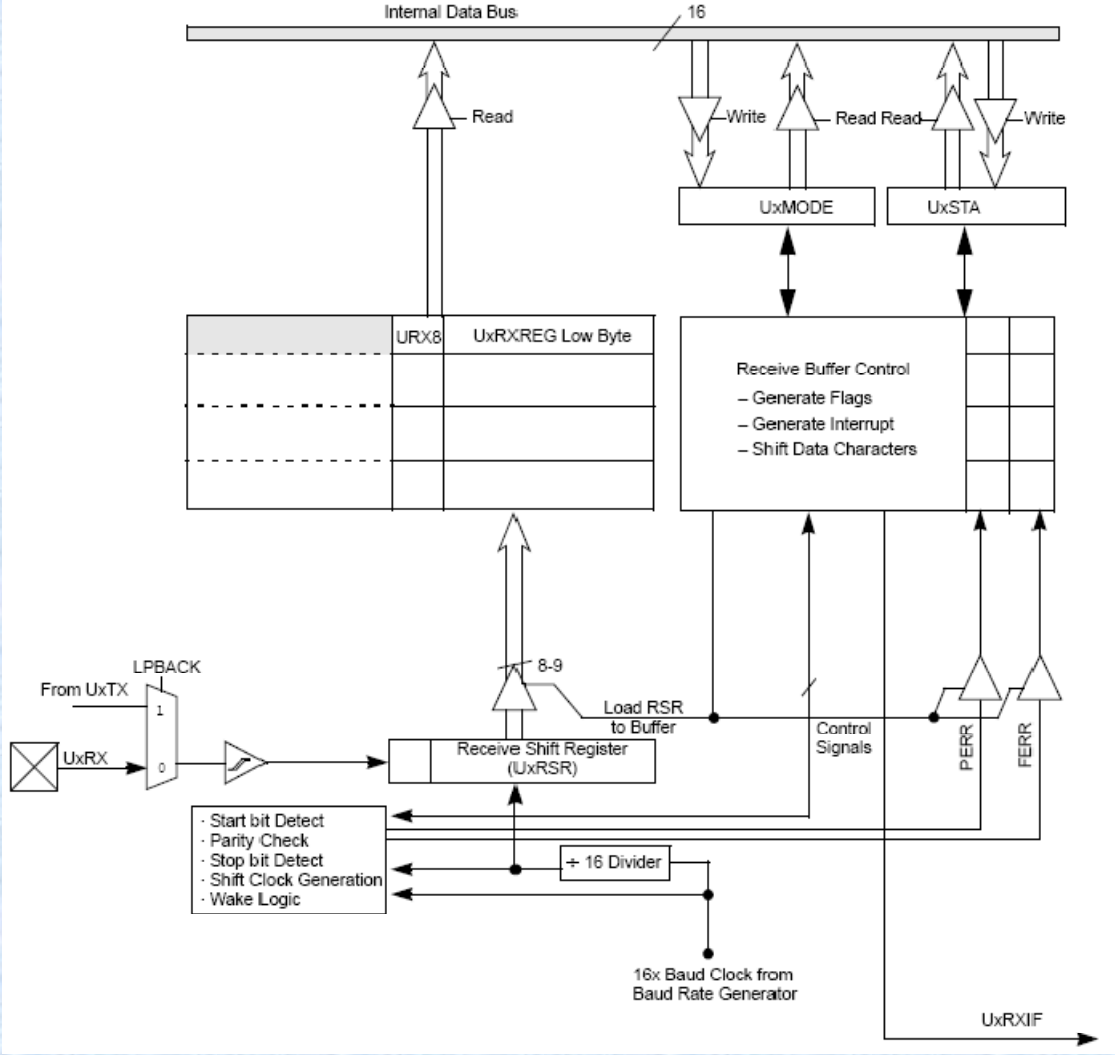
µC dsPIC 30F3010 - UART

- **UART TRANSMITTER BLOCK DIAGRAM**
  - Page 119 du 70141.pdf



µC dsPIC 30F3010 - UART

- **UART RECEIVER BLOCK DIAGRAM**
  - Page 120 du 70141.pdf



µC dsPIC 30F3010 - UART

- **Features (caractéristiques)**
  - Baud Rate = FCY / (16(BRG+1))
  - UxMODE : x =1 seule UART pour le 30F3010

TABLE 18-1: UART1 REGISTER MAP(1)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
U1MODE	020C	UARTEN	—	USIDL	—	—	ALTIO	—	—	WAKE	LPBACK	ABAUD	—	—	PDSEL1	PDSEL0	STSEL
U1STA	020E	UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA
U1TXREG	0210	—	—	—	—	—	—	—	UTX8	Transmit Register							
U1RXREG	0212	—	—	—	—	—	—	—	URX8	Receive Register							
U1BRG	0214	Baud Rate Generator Prescaler															

Legend: u = uninitialized bit; — = unimplemented bit, read as '0'

Note 1: Refer to "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

Exemple d'UART 1/4

```
void InitUART()
{
// Initialize the UART1 for BAUD
          U1MODE = 0x8400;      // enable + alternate pins
//        U1MODE = 0x8000;      // enable + normal pins
          U1STA = 0x0000;
          U1BRG = ((FCY/16)/BAUD) - 1;     // set baud to BAUD (9600 ou 19200 ou…)
          RXPtr = &InData[0];              // point to first char in receive buffer
          Flags.CheckRX = 0;                          // clear rx and tx flags
          IFS0bits.U1RXIF = 0;  // clear interrupt flag
          IEC0bits.U1RXIE = 1;                        // enable RX interrupt
          Flags.SendTX = 0;
          Flags.SendData = 0;              // clear flag
          SeqComm=SeqCommMax;
          U1STAbits.UTXEN = 1;           // Initiate transmission
}
//-----------------------------------------------------------------------
void __attribute__((interrupt, auto_psv)) _U1TXInterrupt(void)
{         IFS0bits.U1TXIF = 0;  }          // clear interrupt flag
//-----------------------------------------------------------------------
void __attribute__((interrupt, auto_psv)) _U1RXInterrupt(void)
{         IFS0bits.U1RXIF = 0;  // clear interrupt flag
          *RXPtr = U1RXREG;
//        U1TXREG = *RXPtr; debug reflect any char received on the RX to the TX
          if (*RXPtr == CR   || *RXPtr == LF || RXPtr>=(&InData[0] +current_inputindexLimit))
                    {
                    Flags.CheckRX = 1;
                    RXPtr = &InData[0];
                    }
          else RXPtr++;
}
```

Exemple d'UART 2/4

```
void SendMsg()
{
while (*TXPtr)
          {
          while (U1STAbits.UTXBF);
          U1TXREG = *TXPtr++;
          }
}
```

➢Déclaration

```
// RS232 ------------------------------------------
unsigned char *TXPtr;
unsigned char *RXPtr;
#define CR 0x0D
#define LF            0x0A
//#define BAUD 19200
#define BAUD 57600    // for BT
unsigned char InData[] = {"00000000000000000000000000000000"};
//unsigned char current_inputindex;
#define current_inputindexLimit 31
unsigned char OutData[]  = {"$RBref=0000 Bmes=0000 Vlim=0000 Vref=0000 :R\r"};
#define OffsetBref 7                        // offset in OutData : position de la val de Bref
#define OffsetBmes 17                       // offset in OutData : position de la val de Bmes
#define OffsetVlim 27                       // offset in OutData : position de la val de Vlim
#define OffsetVref 37                       // offset in OutData : position de la val de Vref
#define Stateoffs  43                       // offset in OutData : position de la val de Running
int SeqComm;                    // ttes les 0.5 s
#define SeqCommMax 500                      // ttes les 0.0625 s
```

Exemple d'UART 3/4

➢Déclaration de la structure des variables de contrôles booléennes

```
struct      {
            unsigned Running      : 1;
            unsigned CheckRX      : 1;
            unsigned SendTX       : 1;
            unsigned SendData     : 1;
            unsigned unused       : 12;
            } Flags;                          // NOTE : Flags n'a rien à voir avec les interruptions
```

•Utilisation : Envoi

```
            TXPtr = &OutData[0];
            SendMsg();
```

Exemple d'UART 4/4

➢Utilisation : Réception

```
//---------------------------------------------------------------------------
void ReceiveData()
{
          TXPtr = &InData[0];
          SendinputMsg();
          Flags.CheckRX=0;
          if (InData[0] != 'C')           return; // sinon, traite la commande
          switch (InData[1])
                    {
                    case 'E' : if (InData[2] == '1') Flags.Running=1;
                                        else       Flags.Running=0;
                               EnableL6234=Flags.Running; // enable or not L6234 principal
                               RunningLED = Flags.Running; // set or clear running LED
                               UpdateBufferRunningStatus(Flags.Running);
                               break;
                    case 'M' : if (InData[2] == '1') { Flags.UARTBref=1;   Bref=BrefUart; }
                                        else       { Flags.UARTBref=0; Bref=ADCBUF0; }
                    case 'F' : if (InData[2] == '1') WriteEEPROM();
                                        else       ReadEEPROM();
                    case 'R' : BrefUart   =ConvHexaToInt( &InData[2]);
                               break;
                    case 'P' : lreg.Field.Kp=ConvHexaToDouble( &InData[2]);
                               break;
                    case 'I' : lreg.Field.Ki=ConvHexaToDouble( &InData[2]);
                               break;
                    case 'D' : lreg.Field.Kd=ConvHexaToDouble( &InData[2]);
                               break;
                    }
}
```

Thèmes non abordés

- **dspic 30F3010**
  - SPI : Serial Peripheral Interface : Liaison série synchrone (avec Clk) rapide
  - I2C : liaison série
  - Bus CAN : liaison série utilisée en industrie (automobile et autres), voir le dspic 30F4012
  - Modes d'économie d'énergie (IDLE et SLEEP)
  - EEPROM : pour garder des informations d'utilisateur (comme les paramètres de réglage d'un PID…) après coupure de l'alimentation
  - QEI : Quadrature Encoder Interface, pour brancher un codeur incrémental de mesure de l'incrément de position et de la vitesse
- **Autres systèmes**
  - DSP : Systèmes numériques de traitement spécialisés, cartes dSPACE. Les principes restent les mêmes que les µC mais ils sont plus puissants
  - API : Automates Programmbles, ne conviennent pas pour la commande rapprochée car ils ne sont pas assez rapides mais ils servent d'interfaces entre les commandes et les superviseurs.