

# Microcontrôleur ATMEL ATMEGA

Les microcontrôleurs de la famille **ATMEGA** en technologie **CMOS** sont des modèles à 8 bits **AVR** basés sur l'architecture **RISC**. En exécutant des instructions dans un cycle d'horloge simple, l'**ATMEGA** réalise des opérations s'approchant de 1 **MIPS** par **MHZ** permettant de réaliser des systèmes à faible consommation électrique et simple au niveau électronique.



**Note :** Une partie des figures de ce document provienne de la documentation officielle **ATMEL** et plus spécialement du modèle **ATMEGA32**. Le reste est de ma conception personnelle.

Un remerciement à **ATMEL** pour leur documentation disponible simplement sur leur site Web.

# Table des Matières

<b>MICROCONTROLEUR ATMEL ATMEGA.....</b>	<b>1</b>
<b>INTRODUCTION .....</b>	<b>7</b>
<b>LA SYNTAXE UTILISEE.....</b>	<b>8</b>
<b>LES MODELES ATMEGA AVR.....</b>	<b>9</b>
<b>SYNOPTIQUE.....</b>	<b>10</b>
<b>L'ARCHITECTURE .....</b>	<b>11</b>
<b>PRESENTATION PHYSIQUE.....</b>	<b>12</b>
DESCRIPTIONS DES BROCHES .....	12
<b>L'HORLOGE SYSTEME ET OPTION .....</b>	<b>13</b>
<i>L'horloge Interne .....</i>	<i>13</i>
<i>L'oscillateur du Timer .....</i>	<i>13</i>
<b>LE PLAN MEMOIRE .....</b>	<b>14</b>
<i>La mémoire programme .....</i>	<i>14</i>
<i>La mémoire de donnée.....</i>	<i>14</i>
<i>La mémoire morte.....</i>	<i>14</i>
<b>TOUS LES REGISTRES.....</b>	<b>15</b>
<b>LES REGISTRES SYSTEMES .....</b>	<b>17</b>
<i>Registre MCUCR (MCU Control Register) .....</i>	<i>17</i>
<i>Registre MCUSR (MCU Control ans Status ).....</i>	<i>17</i>
<i>Registre OSCCAL (Oscilator Calibration Register).....</i>	<i>18</i>
<i>Registre SPMCR (Store Program Memory Control Register).....</i>	<i>19</i>
<b>LES REGISTRES D'ETAT ET DE PILE .....</b>	<b>20</b>
<i>Registre SREG (Status Register).....</i>	<i>20</i>
<i>Registre Pointeur de pile (Stack Pointer).....</i>	<i>20</i>
<b>LE CHIEN DE GARDE OU WATCHDOG (WDT).....</b>	<b>21</b>
<i>Registre WDCTR (Watchdog) .....</i>	<i>21</i>
<i>Mise en marche du Watchdog .....</i>	<i>22</i>
<i>Arrêt du Watchdog .....</i>	<i>22</i>
<b>LES INTERRUPTIONS (INT) .....</b>	<b>23</b>
VECTEURS D'INTERRUPTIONS.....	23
LES REGISTRES D'INTERRUPTION.....	24
<i>Registre GICR (General Interrupt Control Register) .....</i>	<i>25</i>
<i>Registre GIFR (General Interrupt Flag Register).....</i>	<i>25</i>
<i>Registre TIFR (Timer/Counter Interrupt Flag).....</i>	<i>26</i>
<i>Registre TIMSK (Timer/Counter Interrupt Mask).....</i>	<i>26</i>
<b>LA MEMOIRE EEPROM.....</b>	<b>28</b>
L'ACCES EN LECTURE/ECRITURE DANS L'EEPROM .....	28
<i>Registre EEAR (The EEPROM Address Register).....</i>	<i>28</i>
<i>Registre EEDR (The EEPROM Data Register).....</i>	<i>28</i>
<i>Registre EECR (The EEPROM Controle Register).....</i>	<i>29</i>
PROCEDURE DE LECTURE/ECRITURE DANS L'EEPROM .....	29
<i>Procédure d'écriture.....</i>	<i>29</i>
<i>Procédure de lecture .....</i>	<i>29</i>
<i>Exemple de programmation .....</i>	<i>29</i>
<b>LES ENTREES/SORTIES NUMERIQUES (PORTX) .....</b>	<b>30</b>
GENERALITE SUR LES PORTS .....	30
LES REGISTRES DE REFERENCE .....	31
<i>Registre DDRx (Data Direction Register) .....</i>	<i>31</i>

<i>Registre PORTx (Data Register Port)</i> .....	31
<i>Registre PINx (Reading the Pin Value)</i> .....	32
<i>Registre SFIOR (Special Function I/O Register)</i> .....	32
PROGRAMMATION D'UN PORT NUMERIQUE.....	32
<i>Paramétrage d'un port</i> .....	32
<i>Exemple de programmation</i> .....	32
<b>LE COMPAREUR ANALOGIQUE (AC)</b> .....	<b>33</b>
ENTREE MULTIPLEXE DU COMPAREUR ANALOGIQUE.....	33
PRESENTATION DES REGISTRES DU COMPAREUR ANALOGIQUE.....	34
<i>Registre ACSR (Analog Comparator Control and Status Register)</i> .....	34
<i>Registre SFIOR (Special Function I/O Register)</i> .....	34
<i>Registre ADMUX (ADC Multiplexer Selection Register)</i> .....	34
PROGRAMMATION DU COMPAREUR ANALOGIQUE.....	35
<i>Paramétrage du compareur Analogique</i> .....	35
<i>Un exemple de programmation</i> .....	35
<b>LE CONVERTISSEUR ANALOGIQUE / NUMERIQUE (ADC)</b> .....	<b>36</b>
FONCTIONNEMENT DE L'ADC.....	37
PRESENTATION DES REGISTRES DE L'ADC.....	39
<i>Registre ADMUX (ADC Multiplexer Selection Register)</i> .....	39
<i>Registre ADCSRA (ADC Control and Status Register A)</i> .....	41
<i>Registre ADCH et ADCL (ADC Data Register)</i> .....	41
<i>Registre SFIOR (Special Function I/O Register)</i> .....	42
PROGRAMMATION DE L'ADC.....	42
<i>Paramétrage du convertisseur simple</i> .....	42
<i>Exemple de programmation</i> .....	42
<b>LE PRE DIVISEUR DES TIMER/COMPTEUR0 &amp; 1</b> .....	<b>44</b>
SOURCE D'HORLOGE INTERNE.....	44
REMISE A 0 DU PRE DIVISEUR.....	44
SOURCE D'HORLOGE EXTERNE.....	44
<i>Registre SFIOR (Special Function I/O Register)</i> .....	45
<b>LE TIMER/COMPTEUR0 A 8 BITS (TIMER0)</b> .....	<b>46</b>
FONCTIONNEMENT DU TIMER/COMPTEUR0.....	46
LE MODE NORMAL DU TIMER/COMPTEUR0.....	47
LE MODE TIMER A REMISE A 0 SUR COMPARAISON (CTC).....	47
LE MODE MODULATION EN LARGEUR D'IMPULSION RAPIDE (PWM R).....	48
LE MODE PWM CORRECT (PWM C).....	48
LES REGISTRES ASSOCIES AU TIMER/COMPTEUR0.....	49
<i>Registre TCCR0 (Timer/Counter Control Register)</i> .....	49
<i>Registre TCNT0 (Timer/Counter Register)</i> .....	50
<i>Registre OCR0 (Output Compare Register)</i> .....	51
<i>Registre TIFR (Timer/Counter Interrupt Flag)</i> .....	51
<i>Registre TIMSK (Timer/Counter Interrupt Mask)</i> .....	51
PROGRAMMATION DU TIMER/COMPTEUR0.....	51
<i>Paramétrage du TIMER 0</i> .....	51
<i>Exemple de programmation</i> .....	51
<b>LE TIMER/COMPTEUR1 A 16 BITS (TIMER1)</b> .....	<b>53</b>
MODE DE FONCTIONNEMENT DU TIMER1.....	54
LE MODE NORMAL DU TIMER/COMPTEUR1.....	55
LE MODE TIMER1 A REMISE A 0 SUR COMPARAISON (CTC).....	55
LE MODE MODULATION EN LARGEUR D'IMPULSION RAPIDE (PWM).....	56
LE MODE PWM CORRECT.....	57
LE MODE PWM A FREQUENCE CORRIGEE.....	58
LES REGISTRES ASSOCIES AU TIMER/COMPTEUR1.....	59

<i>Registre TCCR1A (Timer/Coounter1 Controle Register A)</i> .....	59
<i>Registre TCCR1B (Timer/Coounter1 Controle Register B)</i> .....	60
<i>Registre TCNT1H &amp; TCNT1L (Timer/Counter1 Register)</i> .....	61
<i>Registre OCR1AH and OCR1AL (Output Compare Register 1 A)</i> .....	61
<i>Registre OCR1BH and OCR1BL (Output Compare Register 1 B)</i> .....	61
<i>Registre ICR1H and ICR1L (Input Compare Register 1)</i> .....	62
<i>Registre TIFR (Timer/Counter Interrupt Flag)</i> .....	62
<i>Registre TIMSK (Timer/Counter Interrupt Mask)</i> .....	62
PROGRAMMATION DU TIMER/COMPTEUR1 .....	63
<i>Paramétrage du TIMER 1</i> .....	63
<i>Exemple de programmation</i> .....	63
<b>LE PRE DIVISEUR DU TIMER/COMPTEUR2 .....</b>	<b>64</b>
<i>Registre SFIOR (Special Function I/O Register)</i> .....	64
<b>LE TIMER/COMPTEUR2 A 8 BITS (TIMER2).....</b>	<b>65</b>
FONCTIONNEMENT DU TIMER/COMPTEUR2 .....	65
LE MODE NORMAL DU TIMER2 .....	66
LE MODE COMPARAISON A REMISE A 0 (CTC) .....	66
LE MODE PWM RAPIDE .....	67
LE MODE PWM CORRECT (PWM C).....	67
LE MODE ASYNCHRONE DU TIMER2 .....	68
LES REGISTRES ASSOCIES AU TIMER/COMPTEUR2 .....	69
<i>Registre TCCR2 (Timer/Coounter2 Controle Register A)</i> .....	70
<i>Registre TCNT2 (Timer/Counter2 Register)</i> .....	71
<i>Registre OCR2 (Output Compare Register 2)</i> .....	71
<i>Registre ASSR (Asynchronous Status Register)</i> .....	71
<i>Registre TIFR (Timer/Counter Interrupt Flag)</i> .....	72
<i>Registre TIMSK (Timer/Counter Interrupt Mask)</i> .....	72
PROGRAMMATION DU TIMER/COMPTEUR2.....	72
<i>Paramétrage du TIMER2</i> .....	72
<i>Exemple 1 Timer2 mode asynchrone</i> .....	72
<i>Exemple 2 : Timer2 mode PWM à 8 bits</i> .....	73
<b>L'INTERFACE SERIE SYNCHRONE SPI.....</b>	<b>74</b>
DESCRIPTION DES BROCHES .....	75
LE FONCTIONNEMENT DE L'INTERFACE SPI.....	75
FONCTIONNALITE DE LA BROCHE SS .....	76
<i>Le mode d'Esclave</i> .....	76
<i>Le mode de Maître</i> .....	76
MODES DE DONNEES .....	76
LES REGISTRES DE L'INTERFACE SPI .....	77
<i>Registre SPCR (SPI Control Register)</i> .....	77
<i>Registre SPSR (SPI Status Register)</i> .....	78
<i>Registre SPDR (SPI Data Register)</i> .....	78
PROGRAMMATION DE LA SPI.....	78
<i>Exemple de transmission SPI</i> .....	78
<i>Exemple de réception SPI</i> .....	79
<b>L'INTERFACE SERIE USART .....</b>	<b>80</b>
GENERATION D'HORLOGE.....	81
<i>Génération d'Horloge Interne - Le Générateur de Vitesse de transmission (en bauds)</i> .....	81
<i>Opération de Vitesse Double (U2X)</i> .....	82
<i>Horloge Externe</i> .....	82
<i>Opération d'Horloge Synchrone</i> .....	82
FORMATS DU MOT DE DONNEE.....	82
<i>Calcul du bit de Parité</i> .....	83

INITIALISATION DE L'USART .....	83
LES REGISTRES DE L'USART.....	83
<i>Registre UCSRA (USART Control and Status Register A)</i> .....	83
<i>Registre UCSRB (USART Control and Status Register B)</i> .....	84
<i>Registre UCSRC (USART Control and Status Register C)</i> .....	84
<i>Registre UDR (USART I/O Data Register)</i> .....	85
<i>Registre UBRR (USART Baud Rate Register)</i> .....	85
PROGRAMMATION DE L'USART .....	87
<i>Paramétrage de l'USART</i> .....	87
<i>Exemple de programmation</i> .....	88
<b>L'INTERFACE I2C (TWI) .....</b>	<b>89</b>
INTERCOMMUNICATION ÉLECTRIQUE .....	89
TRANSFERT DE DONNEES ET FORMAT D'ENCADREMENT .....	89
<i>Transfert de Bit</i> .....	89
CONDITION DE DÉBUT ET D'ARRÊT (START & STOP).....	90
FORMAT DU PAQUET D'ADRESSE.....	90
<i>Appel Général des Esclaves</i> .....	91
FORMAT DU PAQUET DE DONNEES .....	91
TRANSMISSION TYPE.....	91
BUS MULTI-MAÎTRE, ARBITRAGE ET SYNCHRONISATION.....	92
VU D'ENSEMBLE DU MODULE TWI (I2C) .....	94
<i>Broche SCL et SDA</i> .....	94
<i>Unité de Générateur de Taux de Bit</i> .....	94
<i>Unité d'Interface de bus</i> .....	95
<i>Unité de Comparaison d'Adresse</i> .....	95
<i>Unité de Commande</i> .....	95
DESCRIPTION DES REGISTRES TWI.....	95
<i>Registre TWCR (TWI Control Register)</i> .....	95
<i>Registre TWBR (TWI Bit Rate Register)</i> .....	96
<i>Registre TWSR (TWI Status Register)</i> .....	97
<i>Registre TWAR (TWI (Slave) Address Register)</i> .....	97
<i>Registre TWDR (TWI Data Register)</i> .....	97
CODE DES STATUT TWSR.....	98
<i>Mode Transmission Maître avec TWINT = 1</i> .....	98
<i>Mode Réception Maître WTINT = 1</i> .....	99
<i>Mode Réception Esclave WTINT = 1</i> .....	101
<i>Mode Transmission Esclave WTINT = 1</i> .....	102
<i>Code de Statut Indéfini</i> .....	102
PROGRAMMATION DE L'INTERFACE TWI .....	103
<i>Paramétrage de l'interface TWI</i> .....	103
<i>Exemple de programmation</i> .....	104
<b>VERROUILLAGE DES PROGRAMMES ET DES DONNEES .....</b>	<b>105</b>
<i>Verrouillage bit Mémoire (2)</i> .....	105
FUSIBLE DE PROGRAMMATION .....	106
<i>Bit de Fusible haut</i> .....	106
<i>Bit de Fusible bas</i> .....	106
<b>JEU D'INSTRUCTION.....</b>	<b>107</b>
LEGENDE DU JEU D'INSTRUCTION .....	111
<i>Status Register (SREG)</i> .....	111
<i>Registres et Opérandes</i> .....	111
<i>RAMPX, RAMPY, RAMPZ, RAMPD</i> .....	111
<i>EIND</i> .....	111
<i>STACK</i> .....	111



# Introduction

Ce document est, en outre, la traduction de la documentation de l'**ATMEGA** du constructeur **ATMEL**. Pour ma part c'est la solution que j'ai choisie pour apprendre complètement le fonctionnement de ce microcontrôleur qui me semble le plus puissant du marché des microcontrôleurs à 8 bits.

Un autre document est en cours de préparation sur l'initiation au microcontrôleur **ATMEGA** et son assembleur avec le descriptif complet des 131 instructions (voir le document **Atmega32Assembleur.PDF** sur le site Web).

Pour ma part, j'ai commencé la programmation en 1981 avec un ordinateur **ATOM** qui comportait un interpréteur **BASIC** et un assembleur **6502** avec 55 instructions. Cet ordinateur possédait 4 Ko de mémoire **RAM** et 4 Ko de **ROM**, au début, avec une fréquence d'horloge de 1 **MHz**, une interface **K7** et un mode graphique (256x192) en noir & blanc. C'était le tout début de la micro informatique, mais une machine fantastique pour apprendre et mettre en pratique l'électronique numérique et l'informatique. A la même époque les premiers **APPLE** avec le même microprocesseur était mis sur le marché avec le succès qu'on lui connaît.

Par la suite, j'ai utilisé le microprocesseur **6809** de Motorola avec son assembleur plus riche de 138 instructions et une horloge à 4 **MHz**, le quadruple du **6502**, une révolution.

Ensuite je suis passé au microcontrôleur **MC68HC11A1 & F1** de même conception que le **6502**, mais plus puissant puisque intégrant les interfaces de communication et de gestion de port dans une seule puce, le propre d'un microcontrôleur. Le jeu de 144 instructions et une horloge à 8 **MHz** permettait de faire de belle réalisation mais le défaut majeur de ce **CPU** était le manque cruel de mémoire **RAM** interne limité à 1 Ko, il fallait obligatoirement passer par une mémoire externe facile à interfacer mais bloquant 2 port 8 bits, donc moins de ressource disponible et une complication de l'électronique.

La sirène des microcontrôleurs **PIC** est aussi passée par-là, et j'ai commencé à programmer sur le modèle **PIC16F876** qui possède seulement 35 instructions mais cadencé à 20 **MHz** d'horloge. La gestion des 8 Ko de mémoire est très complexe et alourdi considérablement la programmation dès que l'on dépasse les 2 Ko mythiques des **PIC16F84**. De plus le jeu d'instruction est vraiment trop limité pour faire des programmes complexes facilement lisibles. Je n'étais pas satisfait par ce circuit par rapport à mon usage.

J'ai donc recherché un microprocesseur plus puissant mais toujours à 8 bits pour simplifier le circuit électronique et je suis tombé sur l'**AVR** de **ATMEL**, et en visitant le site [www.atmel.com](http://www.atmel.com) ou j'ai trouvé la documentation de la dernière version l'**ATMEGA** qui correspond tout à fait au microcontrôleur que je recherchais, jeu d'instruction complet avec 131 instructions et une fréquence d'horloge de 4 à 16 **MHz**.

L'avantage essentiel est de disposer de plusieurs modèles avec de la mémoire **FLASH** qui passe de 4 **Ko** à 256 **Ko**, ce qui permet de faire évoluer les montages sans problème. La vitesse d'horloge étant de 4 à 16 **MHz** est tout à fait correcte pour la majorité des applications, même très complexe. La présence d'un convertisseur à 10 bits est aussi un atout important comparé au 8 bits du **MC68HC11**, et pour finir, une interface **I2C** nouvelle sur cette gamme, appelé **TWI**.

J'espère que vous trouverez comme moi votre bonheur dans ce document et m'excuse d'avance pour les quelques défauts toujours présents lors d'une traduction pas toujours facile.

Si vous avez des remarques constructives, des commentaires à faire, des questions techniques, je vous invite à les envoyer à mon adresse eMail : [Balade.nono@voila.fr](mailto:Balade.nono@voila.fr).

Mais maintenant il ne vous reste plus qu'à lire, comprendre, tester, et jouer avec les microcontrôleurs **ATMEGA** et ce document.

Vous trouverez à la fin de ce document un index et quelques annexes.

La version 1.2 corrige quelques fautes et la mise en page.

Très bonne lecture à vous !

Jean-Noël, en Janvier 2005 en Gambie.

## La Syntaxe Utilisée

Pour que le document ne soit pas trop indigeste, je l'ai organisé en chapitre et découpé en module simple qui reprenne les grandes fonctions du microcontrôleur (**INT**, **ADC**, **TIMER**, ...). Je n'ai pas cherché à respecter l'ordre de la documentation d'origine, mais plutôt d'entrée en douceur, si c'est réellement possible, dans l'univers du microcontrôleur.

Les fonctions sont marquées en **GRAS** et les instructions sont en *GRAS ITALIQUE*.

La lecture des tableaux des registres doit toujours se faire de gauche à droite, avec les bits de poids fort à gauche '**MSB**' (7, 6, 5 et 4) et les bits de poids faible à droite '**LSB**' (3, 2, 1 et 0). Les fonctions de chaque bit ou groupe de bit sont donc décrites dans le même ordre, poids fort en premier. Donc la première ligne du tableau d'un registre présente les bits dans l'ordre inverse, la deuxième ligne indique l'adresse physique du registre, le nom de la fonction de chaque bit et la troisième ligne permet de savoir si le bit peut être lu (L) et/ou écrit (E).

Le détail des fonctions d'un registre son ensuite décrit avec le rappel de la fonction en **GRAS et en Bleu**, le nom de la fonction d'origine en anglais est en (*GRAS ITALIQUE*) et entre parenthèse et la façon de l'utilisé avec éventuellement un tableau pour les données multiples.

Les données en décimales sont écrites sans signe spécifique, les données en Hexadécimales sont précédées du signe '\$', exemple : **\$10** vaut **16** en décimal, les données en binaires sont précédées d'un **b** et suivie de 8 zéros ou un, exemple : **b00010000** soit **16** en décimal.



# Les Modèles ATMEGA AVR

Vous trouverez dans le tableau qui suit tout les modèles d'ATMEGA existant et à venir (en 2004) :

Product	Flash (KB)	EEPROM (Bytes)	RAM (Bytes)	I/O	SPI	USART	USI	TWI	PWM	On-Chip Debug		10-bit ADC	LCD	Availability
										JTAG	debugWire			
<b>megaAVR</b>														
ATmega 48	4	256	512	23	1	1	-	1	5	-	Y	8	-	Q4-03
ATmega8	8	512	1K	23	1	1	-	1	3	-	-	8	-	Now
ATmega88	8	512	1K	23	1	1	-	1	5	-	Y	8	-	Q4-03
ATmega8515	8	512	512	35	1	1	-	-	3	-	-	-	-	Now
ATmega8535	8	512	512	32	1	1	-	1	4	-	-	8	-	Now
ATmega16	16	512	1K	32	1	1	-	1	4	Y	-	8	-	Now
ATmega162	16	512	1K	35	1	2	-	-	6	Y	-	-	-	Now
ATmega168	16	512	1K	23	1	1	-	1	5	-	Y	8	-	Q4-03
ATmega32	32	1K	2K	32	1	1	-	1	4	Y	-	8	-	Now
ATmega64	64	2K	4K	53	1	2	-	1	8	Y	-	8	-	Now
ATmega128	128	4K	4K	53	1	2	-	1	8	Y	-	8	-	Now
ATmega256	256	4K	8K	53	1	2	-	1	16	-	Y	8	-	Q1-04
<b>LCD AVR</b>														
ATmega169	16	512	1K	53	1	1	Y	-	4	Y	-	8	Y	Now
ATmega329	32	1K	2K	53	1	1	Y	-	4	Y	-	8	Y	Q1-04

Le modèle pris comme référence pour ce document est l'ATMEGA 32-16 PI pour être précis.

La série ATMEGA fournit les particularités suivantes :

- Une **FLASH RAM** de 4 à 256 Ko (Flash Programmable),
- Une **EEPROM** de 256 à 4 Ko (Electrique Ecriture Programmable ROM),
- Une **SRAM** de 512 à 8 Ko octets (Static Random Access Memory),
- De 23 à 53 lignes d'entrée-sortie universelles (Port A, B, C, D, E, F & G),
- 32 registres de travaux universels qui dialoguent directement avec l'unité centrale (**ALU**),
- Une horloge en temps réel (**RTC**),
- Trois Timer/Compteurs flexibles avec comparaison des modes, interruption internes et externes,
- Un comparateur analogique (entrée sur **PB2, PB3**),
- Un convertisseur Analogique/Numérique **ADC** de 8 canaux à 10 bits,
- Un Chien de Garde programmable avec oscillateur interne,
- Une ou deux interfaces série **USART** programmable & périodique (série asynchrone et synchrone),
- Une interface série **SPI** à trois modes sélectionnables (série synchrone),
- Une interface **I2C** pour la gestion d'un bus à 2 fils,
- Une interface **LCD** pour piloter directement un écran **LCD** (modèles spéciaux),
- Tension d'alimentation de 2,7 V à 5,5 V.

Les chapitres qui suivent vous familiariseront avec chacun de ces modules.



# L'Architecture

Le cœur **AVR** combine un jeu de 131 instructions riches avec 32 registres spéciaux travaillant directement avec l'Unité Arithmétique de Logique **ALU**, qui représente le registre d'accumulateur **A** (**B** ou **D**) dans les microcontrôleurs classiques.

Ces registres spéciaux permettent à deux registres indépendants d'être en accès directs par l'intermédiaire d'une simple instruction et exécutée sur un seul cycle d'horloge. Cela signifie que pendant un cycle d'horloge simple l'Unité Arithmétique et Logique **ALU** exécute l'opération et le résultat est stocké en arrière dans le registre de sortie, le tout dans un cycle d'horloge. L'architecture résultante est plus efficace en réalisant des opérations jusqu'à dix fois plus rapidement qu'avec des microcontrôleurs conventionnels **CISC**.

Les registres spéciaux sont dit aussi registre d'accès rapide et 6 des 32 registres peuvent être employés comme trois registre d'adresse 16 bits pour l'adressage indirects d'espace de données (**X**, **Y** & **Z**). Le troisième **Z** est aussi employé comme indicateur d'adresse pour la fonction de consultation de table des constantes.

Les 32 registres sont détaillés dans le tableau qui suit avec l'adresse effective dans la mémoire **SRAM** :

Bit 7 à 0	Adresse	Registre Spéciaux
R0	\$00	
R1	\$01	
Rn	\$xx	
R26	\$1A	Registre <b>X</b> Partie Basse
R27	\$1B	Registre <b>X</b> Partie Haute
R28	\$1C	Registre <b>Y</b> Partie Basse
R29	\$1D	Registre <b>Y</b> Partie Haute
R30	\$1E	Registre <b>Z</b> Partie Basse
R31	\$1F	Registre <b>Z</b> Partie Haute

Les informations sont diffusées par un bus de donnée à 8 bits dans l'ensemble du circuit.

Le microcontrôleur possède aussi un mode sommeil qui arrête l'unité centrale en permettant à la **SRAM**, les Timer/Compteurs, l'interface **SPI** d'interrompre la veille du système pour reprendre le fonctionnement.

Lors de l'arrêt de l'énergie électrique, le mode économie sauve le contenu des registres et gèle l'oscillateur, mettant hors de service toutes autres fonctions du circuit avant qu'une éventuelle interruption logicielle ou matérielle soit émise.

Dans le mode économie, l'oscillateur du minuteur continue à courir, permettant à l'utilisateur d'entretenir le minuteur **RTC** tandis que le reste du dispositif dort.

Le dispositif est fabriqué en employant la technologie de mémoire à haute densité non volatile d'**ATMEL**.

La mémoire **FLASH** est reprogrammable par le système avec l'interface **SPI** ou par un programmeur de mémoire conventionnel non volatile (voir le chapitre sur la programmation du **MPU**).

# Présentation Physique

L'ATMEGA se présente sous la forme d'un circuit intégré à 40 broches pour le modèle **ATMEGA32** en boîtier **PDIP** ou le boîtier **TQFP/MLF** à 44 broches.

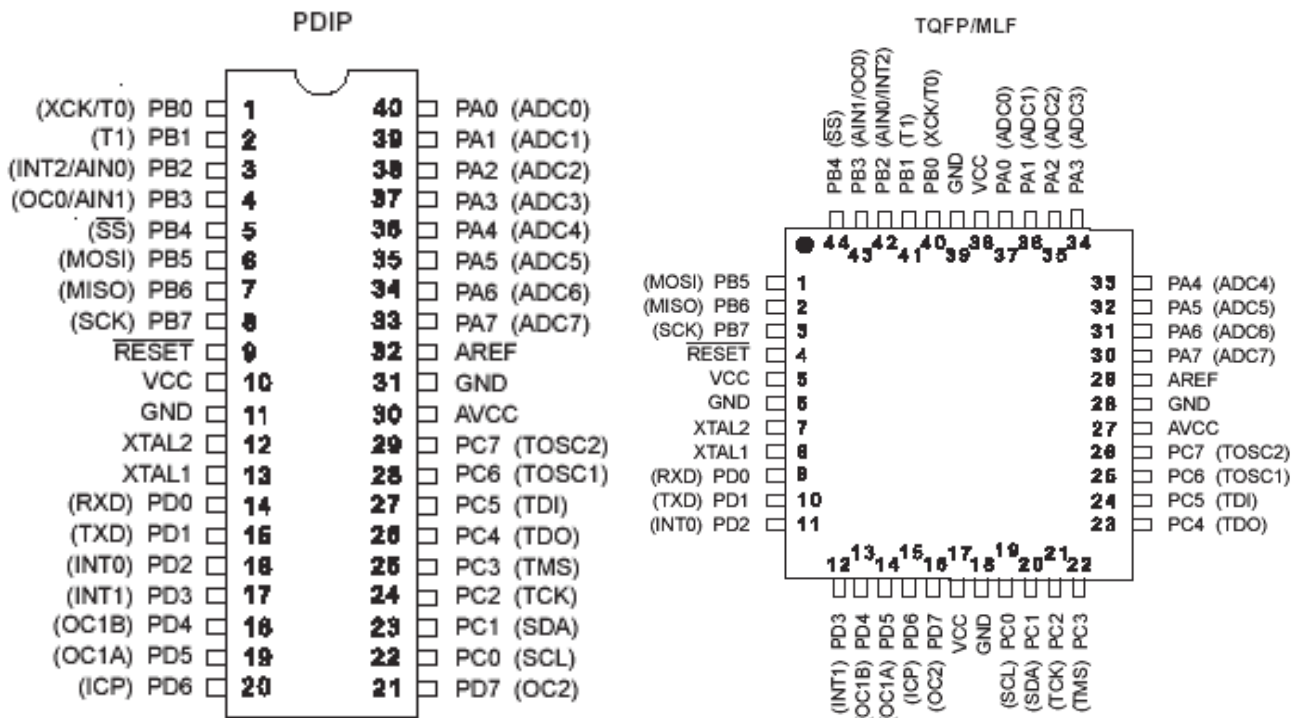


Figure 2, brochage typique d'un ATMEGA32 en boîtier **PDIP** et **PLCC**.

## Descriptions des broches

**Port A (PA7.. PA0)** le Port A est un port d'entrée-sortie à 8 bits bidirectionnel avec des résistances internes de tirage (choisi pour chaque bit). Il sert aussi pour les entrées analogiques du convertisseur A/D. Le Port A (comme le B, C et D) est en position trois états quand une condition de reset devient active, même si l'horloge ne court pas.

**Port B (PB7.. PB0)** le Port B est un port d'entrée-sortie à 8 bits bidirectionnel avec des résistances internes de tirage (choisi pour chaque bit). Il sert aussi de comparateur analogique (sortie sur **PB2**, **PB3**), ou de **SPI**.

**Port C (PC7.. PC0)** le Port C est un port d'entrée-sortie à 8 bits bidirectionnel avec des résistances internes de tirage (choisi pour chaque bit). Il sert aussi comme oscillateur pour le Timer/Compteur2 et d'interface **I2C**.

**Port D (PD7.. PD0)** le Port D est un port d'entrée-sortie à 8 bits bidirectionnel avec des résistances internes de tirage (choisi pour chaque bit). Il sert aussi d'**USART** et d'entrées pour les interruptions externes.

**RESET** déclenché par un front descendant maintenue plus de 50 ns il produira le Reset du microcontrôleur, même si l'horloge ne court pas.

**XTAL1** Entrée de l'oscillateur externe ou libre pour l'horloge interne.

**XTAL2** Production de l'amplificateur d'oscillateur.

**AVCC** est une broche de tension d'alimentation pour le Convertisseur A/D qui doit être connectée à **VCC** via un filtre passe-bas pour éviter les parasites.

**AREF** est l'entrée de référence analogue pour le Convertisseur A/D avec une tension dans la gamme de 2 V à **AVCC** avec filtre passe bas.

**AGND** masse Analogique. Si la masse analogique est séparée de la masse générale, brancher cette broche sur la masse analogique, sinon, connecter cette broche à la masse générale **GND**.

**VCC** broches d'alimentation du microcontrôleur (+3 à +5V).

**GND** masse de l'alimentation.

# L'Horloge Système et Option

Le choix du type d'horloge est déterminé lors de la programmation de la mémoire **FLASH** du microcontrôleur et peut être choisi parmi les données du tableau suivant :

Dispositif d'Option d'horloge	CKSEL3, 2, 1 & 0
Résonateur Externe Cristal/Céramique	1111 - 1010
Cristal Basse Fréquence Externe	1001
Oscillateur Externe à RC	1000 - 0101
Oscillateur Calibré Interne à RC	0100 - 0001
Horloge Externe	0000

L'oscillateur à quartz externe est connecté sur **XTAL1** et **XTAL2** comme le montre la figure suivante pour cadencer le microcontrôleur. La fréquence du quartz est de 4 à 16 **MHz** de type cristal ou un résonateur céramique :

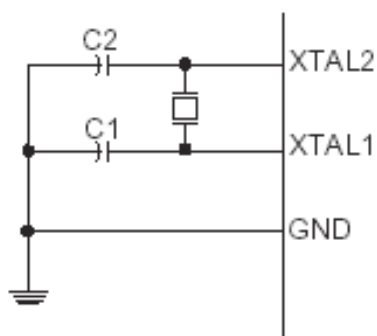


Figure 3. Montage du Quartz.

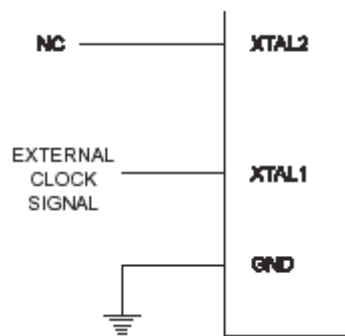


Figure 4. Signale d'Horloge Externe.

**Note :** En employant l'oscillateur de type quartz externe, un amortisseur capacitif de l'ordre de 12 à 22 **pF** doit être connecté comme indiqué dans la figure 3.

## L'horloge Interne

Pour conduire un dispositif externe avec la source d'horloge interne, **XTAL2** doit être laissée en l'air, tandis que l'on connecte **XTAL1** au dispositif à piloter, comme indiqué dans la Figure 4.

## L'oscillateur du Timer

Pour l'oscillateur du Timer temps réel, les broches **TOSC1** et **TOSC2 (PC6 & PC7)** peuvent recevoir un cristal de **32 768 Hz** directement connecté entre ces broches sans condensateur externe. L'application d'une source d'horloge externe sur **TOSC1** n'est pas recommandée. Ce système permet de cadencer le Timer avec des valeurs de temps en sous-multiple de la seconde.

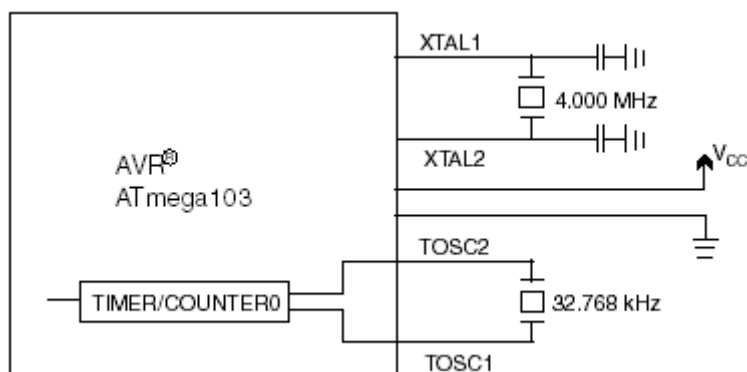


Figure 5, Exemple de connexion d'horloge Temps Réel.

# Le Plan Mémoire

Trois types de mémoire sont utilisés dans la série **ATMEGA**, la mémoire programme **FLASH**, la mémoire de donnée **SRAM** et la mémoire morte de type **EEPROM**.

## La mémoire programme

La mémoire programme permet de stocker et de faire fonctionner le microcontrôleur, il contient de 4 à 256 Ko de programme selon le modèle du microcontrôleur. Le nombre d'écriture sur cette mémoire est limité à 10.000, largement suffisant pour la majorité des applications. La figure 6 donne un exemple de l'adressage de la mémoire **FLASH** du modèle **ATMEGA 32**.

## La mémoire de donnée

La mémoire de donnée contient les 32 registres de travail, les 64 registres de commande et la mémoire **SRAM** pour les variables du programme de 2048 octets pour le modèle **ATMEGA 32**. La figure 7 présente les relations entre espace physique et registre.

## La mémoire morte

La mémoire morte est de type **EEPROM** d'accès plus complexe contiendra la configuration du programme et les données importantes qui seront sauvé pendant l'absence de courant électrique. On peut écrire jusqu'à 100.000 fois dans l'**EEPROM**. La taille de l'**EEPROM** est fonction du modèle de microcontrôleur **ATMEGA** (de 256 bits à 8 Ko).

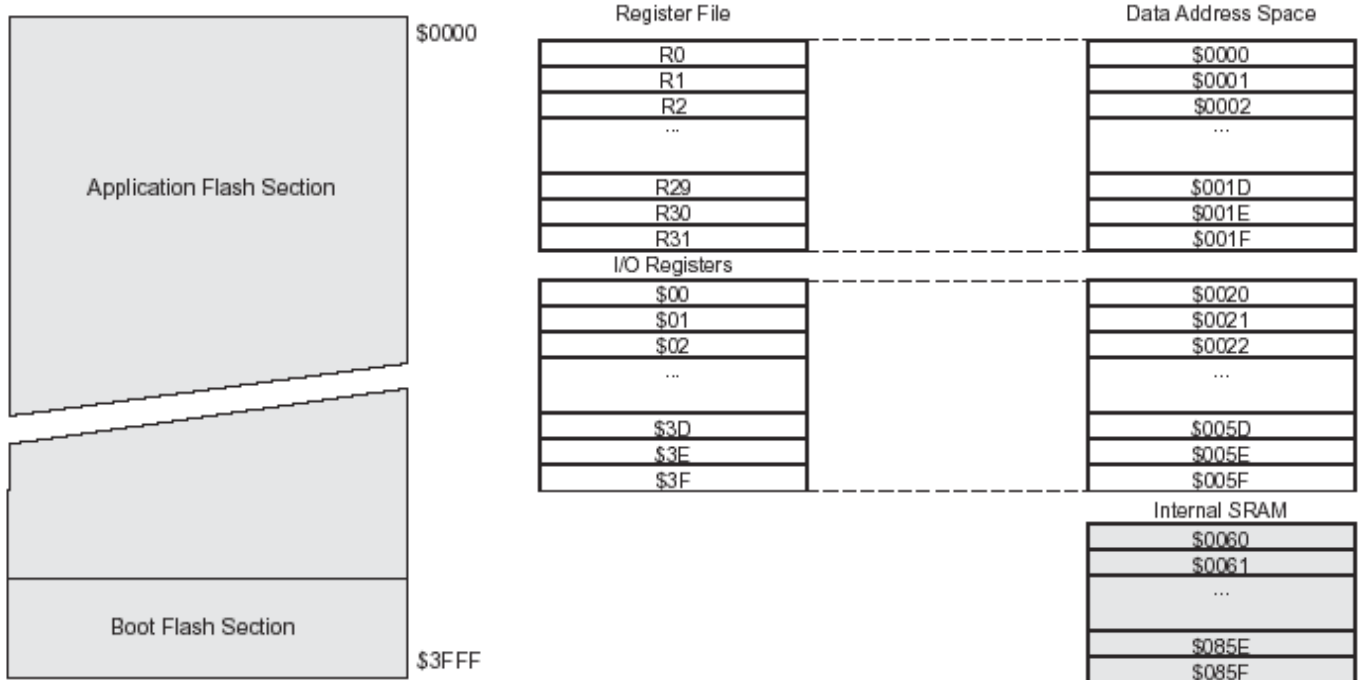


Figure 6, Adressage de la mémoire **FLASH**.

Figure 7, la mémoire de donnée avec la **SRAM**.

# Tous les Registres

Le tableau suivant donne la liste des 66 registres (deux sont communs à d'autres registre) existant dans la gamme ATMEGA :

<b>Catégorie</b>	<b>Désignation</b>	<b>Registre</b>	<b>Adresse</b>
Interface deux fils I2C (TWI)	Contrôle du Bus	<b>TWBR</b>	\$00
	Status	<b>TWSR</b>	\$01
	Adresse	<b>TWAR</b>	\$02
	Données à transmettre ou reçus	<b>TWDR</b>	\$03
Convertisseur analogique	Poids Faible Résultat	<b>ADCL</b>	\$04
	Poids fort Résultat	<b>ADCH</b>	\$05
	Contrôle et statut	<b>ADCSR</b>	\$06
	Sélection de la voie à échantillonner	<b>ADMUX</b>	\$07
Comparateur analogique	Contrôle et statut du Comparateur	<b>ACSR</b>	\$08
USART	Vitesse de communications (Bauds)	<b>UBRR</b>	\$09
	Contrôle	<b>UCR</b>	\$0A
	Status	<b>USR</b>	\$0B
	I/O donnée	<b>UDR</b>	\$0C
SPI	Contrôle	<b>SPCR</b>	\$0D
	Status	<b>SPSR</b>	\$0E
	I/O donnée	<b>SPDR</b>	\$0F
Port parallèle D	Adresse des broches (Pin)	<b>PIND</b>	\$10
	Direction des broches	<b>DDRD</b>	\$11
	Données des broches	<b>PORTD</b>	\$12
Port parallèle C	Adresse des broches (Pin)	<b>PINC</b>	\$13
	Direction des broches	<b>DDRC</b>	\$14
	Données des broches	<b>PORTC</b>	\$15
Port parallèle B	Adresse des broches (Pin)	<b>PINB</b>	\$16
	Direction des broches	<b>DDRB</b>	\$17
	Données des broches	<b>PORTB</b>	\$18
Port parallèle A	Adresse des broches (Pin)	<b>PINA</b>	\$19
	Direction des broches	<b>DDRA</b>	\$1A
	Données des broches	<b>PORTA</b>	\$1B
EEPROM	Contrôle	<b>EECR</b>	\$1C
	Donnée	<b>EEDR</b>	\$1D
	Poids faible adresse	<b>EEARL</b>	\$1E
	Poids Fort adresse	<b>EEARH</b>	\$1F
USART	Vitesse de communications (Bauds)	<b>UBRRH/UCSRC</b>	\$20
Watchdog	Chien de Garde	<b>WDTCR</b>	\$21
Timer / compteur 2 (8 bits)	Statut du mode asynchrone	<b>ASSR</b>	\$22
	Comparaison	<b>OCR2</b>	\$23
	Compteur	<b>TCNT2</b>	\$24
	Contrôle	<b>TCCR2</b>	\$25
Timer / compteur 1 (16 bits)	Entrée poids faible de capture	<b>ICR1L</b>	\$26
	Entrée poids fort de capture	<b>ICR1H</b>	\$27
	Timer B sortie comparaison poids faible	<b>OCR1BL</b>	\$28
	Timer B sortie comparaison poids fort	<b>OCR1BH</b>	\$29
	Timer A sortie comparaison poids faible	<b>OCR1AL</b>	\$2A
	Timer A sortie comparaison poids fort	<b>OCR1AH</b>	\$2B
	Timer/compteur 1 poids Faible	<b>TCNT1L</b>	\$2C
	Timer/compteur 1 poids Fort	<b>TCNT1H</b>	\$2D

<b>Catégorie</b>	<b>Désignation</b>	<b>Registre</b>	<b>Adresse</b>
Port I/O Oscillateur	Contrôle Timer/compteur 1 poids Faible	<b>TCCR1B</b>	\$2E
	Contrôle Timer/compteur 1 poids Fort	<b>TCCR1A</b>	\$2F
	Registre Spéciale des fonctions I/O	<b>SFIOR</b>	\$30
	Calibration d'Oscillateur	<b>OSCCAL/OCDR</b>	\$31
Timer / compteur 0 (8 bits)	Timer/compteur 0	<b>TCNT0</b>	\$32
	Contrôle Timer/compteur 0	<b>TCCR0</b>	\$33
MCU	Statuts Général	<b>MCUSR</b>	\$34
	Contrôle Général	<b>MCUCR</b>	\$35
Interface I2C (TWI)	Contrôle Général	<b>TWCR</b>	\$36
MCU 'Boot Loader'	Contrôle mémoire programme	<b>SPMCR</b>	\$37
Interruptions	Drapeau d'interruption Compteur/Timer	<b>TIFR</b>	\$38
	Masque d'interruption Compteur/Timer	<b>TIMSK</b>	\$39
	Drapeau d'interruption Généraux	<b>GIFR</b>	\$3A
	Masque d'interruption Généraux	<b>GICR</b>	\$3B
	Timer/compteur comparaison en Sortie	<b>OCR0</b>	\$3C
Pointeur de pile	Pointeur de pile poids faible	<b>SPL</b>	\$3D
	Pointeur de pile poids fort	<b>SPH</b>	\$3E
Registre SREG	Registre d'Etat et de Statut	<b>SREG</b>	\$3F

Les registres seront présentés au fur et à mesure de l'avancer du document et des fonctions qui y sont rattachées.



# Les Registres Systèmes

Les registres systèmes permettent de programmer le microcontrôleur selon le choix d'utilisation que le programmeur veut en faire. Ils sont aux nombres de 4, mais seul les deux premiers sont importants. Le fonctionnement du microcontrôleur est modifié lors de la programmation de la mémoire **FLASH**, le choix d'horloge, ...

## Registre MCUCR (*MCU Control Register*)

C'est le registre de contrôle de l'unité centrale du microcontrôleur. Ce registre détermine le fonctionnement en mode sommeil et la gestion des interruptions externe 0 & 1.

Adresse	7	6	5	4	3	2	1	0
\$35	<b>SE</b>	<b>SM2</b>	<b>SM1</b>	<b>SM0</b>	<b>ISC11</b>	<b>ISC10</b>	<b>ISC01</b>	<b>ISC00</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**SE Sleep Enable** Mise en sommeil de l'unité centrale si mis à 1.

**SM2, 1, 0 Sleep Mode** Choix du mode sommeil (arrêt ou ralentie). Les modes de sommeil ne seront pas étudiés mais nous présentons les modes possibles dans le tableau qui suit :

SM2	SM1	SM0	Mode Sommeil (Sleep)
0	0	0	Fonctionnement Normal
0	0	1	ADC à Réduction Bruit
0	1	0	Arrêt Alimentation
0	1	1	Alimentation Sauvé
1	0	0	Réservé
1	0	1	Réservé
1	1	0	En Pause
1	1	1	En Pause Etendue

**ISC11, ISC10 Interrupt Sense Control**, définition de prise en compte de l'interruption externe **INT1\***.

**ISC01, ISC00 Interrupt Sense Control**, définition de prise en compte de l'interruption externe **INT0\***.

\* Tableau récapitulatif des modes de programmation **INT0** et **INT1** :

ISC01	ISC00	Mode de Déclenchement	ISC11	ISC10
0	0	Sur Niveau Bas	0	0
0	1	Sur Changement de Niveau (0/1 ou 1/0)	0	1
1	0	Sur Front descendant	1	0
1	1	Sur Front montant	1	1

## Registre MCUSR (*MCU Control and Status*)

Ce registre permet de connaître le statut du microcontrôleur, comme de déterminer la source d'un reset, ...

Adresse	7	6	5	4	3	2	1	0
\$34	<b>JTD</b>	<b>ISC2</b>	-	<b>JTRF</b>	<b>WDRF</b>	<b>BORF</b>	<b>EXTRF</b>	<b>PORF</b>
L/E	L/E	L/E	L	L/E	L/E	L/E	L/E	L/E

Sur l'ATMEGA, le bit 5 est inutilisé.

**JTD JTAG Interface Disable** Quand ce bit est à 0, on permet l'interface **JTAG** si le bit **JTAGEN** est à 1. Si ce bit est à 1, l'interface **JTAG** est mise hors de service. **Attention** : ne pas manipuler ce bit, il peut bloquer la programmation de la mémoire **FLASH**. **Note** : **JTAG** = Interface de programmation de l'ATMEGA.

**ISC2 Interrupt Sense Control 2** Il est activé par la broche externe **INT2** si le registre **SREG** est activé pour les interruptions (**I** = 1). Si **ISC2** est à 0, un front descendant sur **INT2** active l'interruption. Si **ISC2** est à un, un front montant sur **INT2** active l'interruption. Les fronts sur **INT2** sont enregistrés d'une manière asynchrone. Les impulsions sur **INT2** plus large que la largeur d'impulsion minimum de 50 ns produiront une interruption, celle inférieure ne sont pas

garantit. En changeant le bit **ISC2** une interruption peut arriver, donc, on recommande de mettre d'abord hors service **INT2** en désactivant le bit **GICR**. Finalement, **INT2** doit être réarmé avant que l'interruption ne reprenne.

**JTRF JTAG Reset Flag** Ce bit est mis à 1 si un Reset est causé par la logique **JTAG** du Registre Reset choisi par l'instruction **JTAG AVR\_RESET**. Ce bit est remis à 1 lors de la mise sous tension, ou en écrivant un 0 dedans.

**WDRF Watchdog Reset Flag** Est mis à 1 pour activer le chien de garde. Le bit est remis à 0 par un Reset ou par l'écriture d'un 0.

**BORF Brown-out Reset Flag** Ce bit est mis à 1 si une Panne d'électricité partielle arrive. Le bit est remis par un Reset ou par l'écriture d'un 0.

**EXTRF & PORF External Reset Flag & Power-on Reset Flag** Détermine la source d'un **RESET**, le décodage peut se résumer ainsi :

Source du Reset	EXTRF	PORF
WatchDog	0	0
Mise Sous Tension	0	1
Reset Externe	1	0
Mise Sous Tension	1	1

### Registre OSCCAL (Oscillator Calibration Register)

Le registre de calibrage de l'oscillateur pour la programmation de la mémoire **FLASH** et l'**EEPROM**. Les valeurs par défaut conviennent pour la majorité des applications.

Adresse	7	6	5	4	3	2	1	0
\$31	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
L/E	L/E	L/E	L	L/E	L/E	L/E	L/E	L/E

**CAL7 à 0 Oscillator Calibration Value** L'écriture de l'octet de calibrage à cette adresse coupera l'oscillateur interne pour enlever les variations de fréquence de l'oscillateur. Pendant le **RESET** la valeur de calibrage à une fréquence de 1 **MHz** est placée dans l'adresse \$00 puis est automatiquement chargée dans le registre d'**OSCCAL**. Si un oscillateur de type **RC** interne est employé à d'autres fréquences, les valeurs de calibrage doivent être chargées manuellement. Cela peut être fait par la première lecture à l'adresse \$00 par un programme et en stockant ensuite les valeurs de calibrage dans la mémoire **FLASH** ou l'**EEPROM**. Alors la valeur peut être lue par le logiciel et chargée dans le registre d'**OSCCAL**. Quand **OSCCAL** est à zéro, la fréquence disponible la plus basse est choisie. L'écriture d'une valeur différente de zéro dans ce registre augmentera la fréquence de l'oscillateur interne. L'écriture de **\$FF** dans le registre donnera la fréquence disponible la plus haute. L'oscillateur calibré est utilisé pour donner la base de temps à la logique de lecture/écriture de l'**EEPROM** et l'accès à la mémoire **FLASH**. Si l'**EEPROM** ou la mémoire **FLASH** est déjà écrite, ne calibrez pas à plus de 10 % au-dessus de la fréquence nominale. Autrement, l'écriture dans l'**EEPROM** ou la **FLASH** peut échouer. Notez que l'oscillateur est destiné au calibrage des fréquences de 1.0, 2.0, 4.0, ou 8.0 **MHz**. On ne garantit pas le réglage d'accord à d'autres valeurs que ceux indiqués ci-dessous.

OSCCAL	Minimum de la Fréquence Nominal (%)	Maximum de la Fréquence Nominal (%)
\$00	50	100
\$7F	75	150
\$FF	100	200

## Registre SPMCR (*Store Program Memory Control Register*)

Le registre de contrôle de sauvegarde de la mémoire de programme (SPM) contient les bits pour contrôler les opérations de chargement des programmes dans la mémoire FLASH en mode 'Boot'. Ce registre est donné à titre indicatif seulement. Voir la documentation ATMEL pour plus d'information.

Adresse	7	6	5	4	3	2	1	0
\$37	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEM
L/E	L/E	L/E	L	L/E	L/E	L/E	L/E	L/E

Sur l'ATMEGA, le bit 5 est inutilisé.

**SPMIE** *SPM Interrupt Enable* Quand le bit est écrit à 1 et le bit **I** de **SREG** est mis à 1, le **SPM** est prêt à prendre les interruptions. L'interruption **SPM** est exécutée tant que le bit **SPMEN** est à 0.

**RWWSB** *Read-While-Write Section Busy* Quand une auto programmation (Page Effacée ou Page Écrite) de la section **RWW** est initialisé, le bit **RWWSB** sera mis à 1 par le matériel. Quand le bit **RWWSB** est mis à 1, la section **RWW** ne peut pas avoir accès à la mémoire. Le bit **RWWSB** sera remis à 0 si le bit **RWWSRE** est écrit à 1 après une opération d'auto programmation achevée. Alternativement le bit **RWWSB** sera automatiquement remis à 0 si une opération de changement de page est initialisée.

**RWWSRE** *Read-While-Write Section Read Enable* Quand la programmation (Page Effacée ou Page Écrite) de la section **RWW**, la section **RWW** est bloquée pour la lecture (le **RWWSB** sera mis à 1 par le matériel). Pour autorisé à nouveau la section **RWW**, le logiciel d'utilisateur doit attendre que la programmation soit achevée (**SPMEN** remis à 0). Alors, si le bit **RWWSRE** est écrit à un en même temps que **SPMEN**, l'instruction suivante **SPM** après quatre cycles d'horloge autorise à nouveau la section **RWW**. On ne peut pas autorise de nouveau la section **RWW** pendant que la mémoire **FLASH** est occupée (**SPMEN** actif). Si le bit **RWWSRE** est écrit tandis que la **FLASH** est chargé, l'opération de chargement de la mémoire **FLASH** échouera et les données chargées seront perdues.

**BLBSET** *Boot Lock Bit Set* Si ce bit est écrit à un en même temps que **SPMEN**, l'instruction suivante **SPM** après quatre cycles d'horloge, sera accordé avec les données dans **R0**. Les données dans **R1** et l'adresse dans le registre **Z** sont ignorées. Le bit **BLBSET** sera automatiquement remis à 0 à l'achèvement de l'instruction ou si aucune instruction **SPM** n'est exécutée dans les quatre cycles d'horloge.

**PGWRT** *Page Write* Si ce bit est écrit à un en même temps que **SPMEN**, l'instruction suivante **SPM** après quatre cycles d'horloge est exécutée, la page est écrite avec les données stockées dans le registre provisoire. L'adresse de page est prise dans la partie haute du registre **Z**. Les données dans **R1** et **R0** sont ignorées. Le bit **PGWRT** est remis à 0 automatiquement sur l'achèvement d'une page écrite ou si aucune instruction **SPM** n'est exécutée après quatre cycles d'horloge. L'UC est interrompue pendant l'opération d'écriture de la page entière si la section **NRWW** est adressée.

**PGERS** *Page Erase* Si ce bit est écrit à un en même temps que **SPMEN**, l'instruction suivante **SPM** après quatre cycles d'horloge est exécutée, la page est écrite avec les données stockées dans le registre provisoire. L'adresse de page est prise dans la partie haute du registre **Z**. Les données dans **R1** et **R0** sont ignorées. Le bit **PGWRT** est remis à 0 automatiquement sur l'achèvement d'une page écrivent, ou si aucune instruction **SPM** n'est exécutée après quatre cycles d'horloge. L'UC est interrompue pendant l'opération d'écriture de la page entière si la section **NRWW** est adressée.

**SPMEN** *Store Program Memory Enable* Ce bit permet à l'instruction **SPM** d'attendre quatre cycles d'horloge pour écrire un ensemble ou bien avec les bits **RWWSRE**, **BLBSET**, **PGWRT** ou encore **PGERS**. L'instruction suivant **SPM** aura une signification spéciale, si seulement **SPMEN** est écrit, l'instruction suivant **SPM** stockera la valeur dans **R1** et **R0** dans la zone provisoire adressé par le registre **Z**. Le **LSB** (poids faible) du registre **Z** est ignoré. Le bit **SPMEN** est remis à 0 automatiquement sur l'achèvement d'une instruction **SPM** ou si aucune instruction **SPM** n'est exécutée dans les quatre cycles d'horloge.

# Les Registres d'Etat et de Pile

Deux registres sont indispensables au système pour fonctionner, le registre d'état **SREG** et le registre de gestion de la pile pour les interruptions et la gestion des sous-programmes.

## Registre SREG (*Status Register*)

Le registre **SREG** ou registre d'état sert principalement avec les fonctions arithmétiques et logiques pour les opérations de branchements. Il indique et autorise aussi le fonctionnement des interruptions. Il est modifié par le résultat des manipulations mathématiques et logiques. C'est le principal registre qui sert à effectuer les branchements conditionnels après une opération arithmétique ou logique. Il peut être lu et écrit à tout moment sans aucune restriction.

Par exemple :  $6-8 = -2$ , le bit de Négatif (N) sera mis à 1 (Voir le tableau des instructions).

Adresse	7	6	5	4	3	2	1	0
\$3F	<b>I</b>	<b>T</b>	<b>H</b>	<b>S</b>	<b>V</b>	<b>N</b>	<b>Z</b>	<b>C</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**I** *Global Interrupt enable* sert à activer (1) ou interdire (0) toutes les sources d'interruptions. Si ce bit n'est pas activé alors que vous avez programmé des interruptions, elles ne seront pas présent en compte.

**T** *Copy Storage* joue un rôle de tampon lors de manipulation de bits avec les instructions **BLD** et **BST**.

**H** *Half Cary* signale qu'une demie-retenue à été réalisé lors de l'emploi d'une instruction arithmétique.

**S** *Sign bit* bit de signe résultant d'un OU exclusif avec le bit **N** et **V**.

**V** *Overflow bit* indique un dépassement de capacité lors des opérations arithmétique et logique.

**N** *Negative bit* signale que le résultat de la dernière manipulation arithmétique est négatif.

**Z** *Zéro bit* le résultat de la dernière manipulation arithmétique est égal à zéro.

**C** *Cary bit* l'opération arithmétique à donné lieu à une retenue.

## Registre Pointeur de pile (*Stack Pointer*)

Le registre de pointeur de pile est utilisé par les instructions **PUSH** et **POP** de gestion de pile. La gestion de pile utilise les deux registres 8 bits qui suivent pour former une adresse 16 bit :

Adresse	15	14	13	12	11	10	9	8
\$3D	<b>SP15</b>	<b>SP14</b>	<b>SP13</b>	<b>SP12</b>	<b>SP11</b>	<b>SP10</b>	<b>SP9</b>	<b>SP8</b>
Adresse	7	6	5	4	3	2	1	0
\$3E	<b>SP7</b>	<b>SP6</b>	<b>SP5</b>	<b>SP4</b>	<b>SP3</b>	<b>SP2</b>	<b>SP1</b>	<b>SP0</b>

L'adresse ainsi crée doit être positionné en général en fin de la mémoire **SRAM**. La pile est décrémenté avec l'instruction **PUSH** et incrémenté avec **POP**, donc le positionnement en fin de **SRAM** ne pose aucun problème de taille en général, mais attention aux boucles trop grande avec des **PUSH** qui pourrait arriver dans l'espace de stockage des variables programmes ou système.

**Attention :** Le pointeur de pile est initialisé à la valeur \$0060, il faut changer immédiatement son pointeur en début de programme en le positionnant en fin de mémoire **SRAM** en général !

# Le Chien de Garde ou Watchdog (WDT)

Le chien de garde ou Watchdog, est un compteur qui permet de palier au blocage du microcontrôleur. Ce blocage peut être d'ordre logiciel (retour impossible, mis en boucle infinie ou tout simplement erreur de structuration), soit matériel (parasites, chute de tensions) ; dans les deux cas, le blocage du programme peut avoir des conséquences très embêtantes.

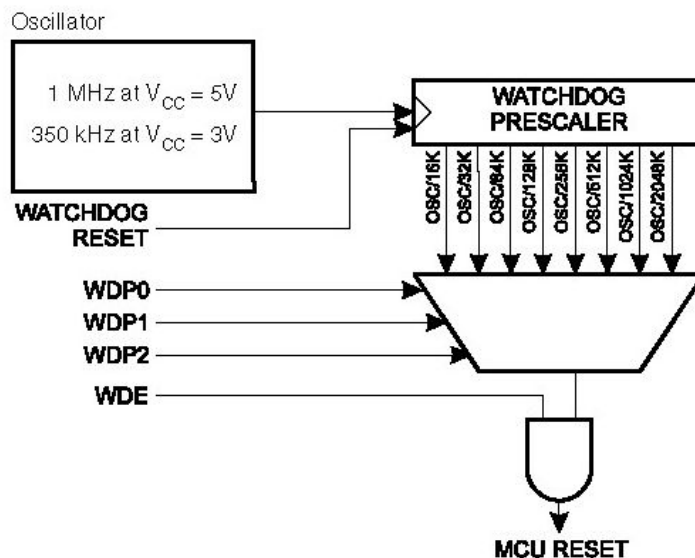


Figure 8, Synoptique du pré diviseur du Watchdog.

Le Watchdog est un compteur dont la source d'horloge est indépendante du quartz du microcontrôleur, mais pas de l'alimentation. La fréquence de fonctionnement est de 1 MHz pour une tension d'alimentation de 5 V et 350 KHz pour 3 V.

Le compteur part de 0, incrémente de 1 à chaque top d'horloge et reviens à 0 après la valeur 255 et on recommence, tout débordement du compteur (255 -> 0) génère un Reset automatique.

A vous de prévoir périodiquement une remise à zéro du compteur par l'instruction assembleur 'WDR', par exemple dans la boucle de gestion du programme principal. Vous pouvez à tout moment déterminer la source d'un RESET grâce au registre MCUSR vu dans les registres systèmes.

## Registre WDCTR (Watchdog)

C'est le registre de contrôle du Watchdog. La mise en marche du Watchdog doit se faire le plutôt possible, ensuite placez l'instruction WDR de manière à assurer un déroulement correct du programme et faite attention aux temporisations, boucles d'attente de conditions et les interruptions.

Adresse	7	6	5	4	3	2	1	0
\$41	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0
L/E	L	L	L	L/E	L/E	L/E	L/E	L/E

Sur l'ATMEGA, le bit 7, 6 & 5 sont inutilisés.

**WDTOE Watchdog Trun-On Enable** Bit de sécurité d'arrêt du chien de garde. Une sécurité logique permet de ne pas l'arrêter accidentellement. Il faut respecter la procédure suivante : **WDTOE** = 1, puis aussitôt, **WDE** = 0. Si l'opération n'est pas effectuée dans les 4 prochains cycles d'horloge, ce bit repasse automatiquement à 0.

**WDE Watchdog Enable** Marche/Arrêt du Watchdog. La désactivation de ce bit n'est possible qu'avec la procédure précédente.

**WDP2, 1, 0 Watchdog Timer Prescaler2, 1, 0** Sélection du facteur de pré division du signal d'horloge, en fait vous définissez l'intervalle maximum de temps pour la remise à 0 RAZ. L'oscillateur interne étant entièrement dépendant de la tension d'alimentation, le temps maximum entre deux RAZ varie en fonction des paramètres des trois bits **WDPx** et de la tension d'alimentation du microcontrôleur, voir le tableau suivant :

WDP2	WDP1	WDP0	Temps Max entre 2 RAZ, VCC= 3V	Temps Max entre 2 RAZ, VCC= 5V
0	0	0	47 ms	15 ms
0	0	1	94 ms	30 ms
0	1	0	190 ms	60 ms
0	1	1	380 ms	120 ms
1	0	0	750 ms	240 ms
1	0	1	1500 ms	490 ms
1	1	0	3000 ms	970 ms
1	1	1	6000 ms	1900 ms

### Mise en marche du Watchdog

```

Wdr                ; Mis à 0 compteur
Ldi  r16,$0F       ; WDE = 1 avec base de temps maximum
Out  WDTCR,r16

```

### Arrêt du Watchdog

```

Ldi  r16,$1E       ; WDTOE et WDE = 1
Out  WDTCR,r16
Ldi  r16,$10       ; WDTOE =1 et WDE = 0
Out  WDTCR,r16

```

# Les Interruptions (INT)

La gestion des interruptions paraît souvent difficile et pourtant ce n'est pas plus compliqué que de gérer une interface **SPI** par exemple.

Afin d'éviter les boucles d'attente sans fin et les risques de blocage, il est assez simple de mettre en œuvre les interruptions du microcontrôleur qui dispose d'un grand nombre de possibilité d'activation.

En premier lever le drapeau d'autorisation d'interruption général dans **SREG** (instruction **SEI**) et il faut donner au périphérique concerné la possibilité d'interrompre provisoirement le programme principal en réglant les bits d'interruption dans les registres respectifs.

Chaque interruption va provoquer un saut de programme à une adresse bien précise correspondant à votre programme de gestion de l'interruption que vous avez écrit (voir le tableau décrit plus loin).

Utiliser l'instruction **RETI** pour effectuer la fin de l'interruption et provoqué le retour au programme principal.

Sauvegarder impérativement vos registres de travail, surtout **SREG**, dans la pile ou en mémoire, car dans le programme d'interruption, la valeur de ce registre à de très grande chance de changer et au retour de l'interruption vous pouvez avoir de drôle de surprise. Pour sauvegarder un registre, il faut le mettre sur la pile avec l'instruction **PUSH**, au début du programme d'interruption, puis le replacer à l'aide de l'instruction **POP** en fin de programme avant **RETI**.

Si une seconde interruption intervient pendant le traitement de la première, le programme la traitera aussitôt fini la première (au retour sur **RETI**).

**Attention :** les interruptions peuvent bloquer l'UC si une boucle infinie a été écrite par erreur, vérifier le système de bouclage dans le programme de gestion des interruptions, il doit se terminer obligatoirement par une sortie **RETI** quelles que soient les conditions testées dans la gestion des interruptions.

## **Vecteurs d'interruptions**

Les vecteurs d'interruptions sont en fait les adresses de correspondant des programmes de gestion des interruptions proprement dit classé par ordre décroissant d'important.

<b>Vecteur</b>	<b>Adresse</b>	<b>Programme</b>	<b>Définition de la Source de l'Interruption</b>
1	\$000	<b>RESET</b>	Reset Externe, Début d'Alimentation, Reset Erreur d'Alimentation, Watchdog Reset, et Reset <b>JTAG AVR</b>
2	\$002	<b>INT0</b>	Interruption Externe 0
3	\$004	<b>INT1</b>	Interruption Externe 1
4	\$006	<b>INT2</b>	Interruption Externe 2
5	\$008	<b>TIMER2 COMP</b>	Comparaison Timer/Compteur2
6	\$00A	<b>TIMER2 OVF</b>	Débordement Timer/Compteur2
7	\$00C	<b>TIMER1 CAPT</b>	Événement de Capture du Timer/Compteur1
8	\$00 <sup>E</sup>	<b>TIMER1 COMPA</b>	Comparaison A Timer/Compteur1
9	\$010	<b>TIMER1 COMPB</b>	Comparaison B Timer/Compteur1
10	\$012	<b>TIMER1 OVF</b>	Débordement Timer/Compteur1
11	\$014	<b>TIMER0 COMP</b>	Comparaison Timer/Compteur0
12	\$016	<b>TIMER0 OVF</b>	Débordement Timer/Compteur0
13	\$018	<b>SPI, STC</b>	Transfert Série Complet
14	\$01A	<b>USART, RXC USART,</b>	Réception Rx Complet
15	\$01C	<b>USART, UDRE USART</b>	Registre de Donnée Vide
16	\$01E	<b>USART, TXC USART,</b>	Transmission Tx Complet
17	\$020	<b>ADC</b>	<b>ADC</b> Conversion Complet
18	\$022	<b>EE_RDY</b>	<b>EEPROM</b> Prêt
19	\$024	<b>ANA_COMP</b>	Comparsateur Analogique
20	\$026	<b>TWI</b>	Interface Série I2C (TWI)
21	\$028	<b>SPM_RDY</b>	Programme de Stockage de la Mémoire Prêt

**Notes : 1** Quand le bit **BOOTRST** est programmé, le dispositif sautera à l'adresse du « Boot Loader » automatiquement.

**Note : 2** Quand le bit **IVSEL** dans le registre **GICR** est mis à 1, les vecteurs interruptions seront déplacés au début de la section de la mémoire **FLASH** du « Boot ». L'adresse de chaque vecteur d'interruption sera alors l'adresse dans cette table supplémentaire à l'adresse de début de la section de la mémoire **FLASH** de « Boot ».

Le **RESET** à la priorité la plus grande. Un programme doit correspondre à chaque vecteur d'interruption, si vous n'utilisez pas l'interruption un branchement sur un programme générique est conseillé, celui-ci peut simplement contenir l'instruction **RETI**.

Tous les programmes devront contenir au début, la déclaration des vecteurs d'interruption comme le montre l'exemple suivant :

Adresse	Labels	Code	Commentaires
.org \$0000			; Positionnement au début de la mémoire
\$0000	jmp	RESET	; Reset Handler
\$0002	jmp	EXT_INT0	; IRQ0 Handler
\$0004	jmp	EXT_INT1	; IRQ1 Handler
\$0006	jmp	EXT_INT2	; IRQ2 Handler
\$0008	jmp	TIM2_COMP	; Timer2 Compare Handler
\$000A	jmp	TIM2_OVF	; Timer2 Overflow Handler
\$000C	jmp	TIM1_CAPT	; Timer1 Capture Handler
\$000E	jmp	TIM1_COMPA	; Timer1 CompareA Handler
\$0010	jmp	TIM1_COMPB	; Timer1 CompareB Handler
\$0012	jmp	TIM1_OVF	; Timer1 Overflow Handler
\$0014	jmp	TIM0_COMP	; Timer0 Compare Handler
\$0016	jmp	TIM0_OVF	; Timer0 Overflow Handler
\$0018	jmp	SPI_STC	; SPI Transfer Complete Handler
\$001A	jmp	USART_RXC	; USART RX Complete Handler
\$001C	jmp	USART_UDRE	; UDR Empty Handler
\$001E	jmp	USART_TXC	; USART TX Complete Handler
\$0020	jmp	ADC	; ADC Conversion Complete Handler
\$0022	jmp	EE_RDY	; EEPROM Ready Handler
\$0024	jmp	ANA_COMP	; Analog Comparator Handler
\$0026	jmp	TWI	; Two-wire Serial Interface Handler
\$0028	jmp	SPM_RDY	; Store Program Memory Ready Handler
			;
\$002A	RESET:	ldi r16,high(RAMEND)	; Début du programme système
\$002B		out SPH,r16	; Début du pointeur de pile en fin de SRAM
\$002C		ldi r16,low(RAMEND)	
\$002D		out SPL,r16	
\$002E		sei	; Activation des interruptions
\$002F	DEBUT	<instr> xxx	; Début du programme utilisateur.
		...	...

## Les registres d'Interruption

Quatre registres d'interruption existent dans l'**ATMEGA**, ils vous sont présentés maintenant.



## Registre GICR (*General Interrupt Control Register*)

Le registre **GICR** contrôle l'activation des interruptions externes et le déplacement de la table des vecteurs d'interruption dans la mémoire. Le registre **GIFR** permet de connaître l'auteur de l'interruption **INTx**.

Adresse	7	6	5	4	3	2	1	0
\$3B	<b>INT1</b>	<b>INT0</b>	<b>INT2</b>	-	-	-	<b>IVSEL</b>	<b>IVCE</b>
L/E	L/E	L/E	L/E	L	L	L	L/E	L/E

Les deux premiers bits (0 & 1) sont réservés à des applications spéciales et complexes du microcontrôleur qui ne seront pas détaillés ici. Les bits 2, 3 et 4 ne sont pas utilisés (lu à 0).

**INT1, 0, External Interrupt Request 1 ou 0 Enable** Si ce bit est mis à 1 l'interruption externe est activée. Les bits **ISC11** et **ISC10** dans le registre de contrôle **MCUCR** définissent si le déclenchement se fait sur un front montant ou descendant, même si le port est configuré en sortie.

**INT2, External Interrupt Request 2 Enable** Si ce bit est mis à 1 l'interruption externe est activée. Le bit **ISC2** dans le registre de contrôle **MCUCSR** définit si le déclenchement se fait sur un front montant ou descendant, même si le port est configuré en sortie.

**IVSEL Interrupt Vector Select** Quand ce bit est à 0, les vecteurs d'interruption sont placés au début de la mémoire **FLASH**. Quand ce bit est mis à 1, les vecteurs d'interruption sont déplacés au début de la section du « Boot Loader » de la mémoire **FLASH**. **Attention** : ne pas modifier ce bit dans votre programme, il doit rester à 0.

**IVCE Interrupt Vector Change Enable** Le bit **IVCE** doit être écrit à 1 pour permettre le changement du bit **IVSEL**, c'est une protection pour éviter de bloquer le microcontrôleur. **IVCE** est remis à 0 par le système quatre cycles d'horloge après sa modification ou quand **IVSEL** est écrit. **Attention** : ne pas modifier ce bit dans votre programme, il doit rester à 0.

## Registre GIFR (*General Interrupt Flag Register*)

Le registre **GIFR** indique l'état des interruptions externes broches **PD2 (INT0)**, **PD3 (INT1)** et **PB2 (INT2)**.

Adresse	7	6	5	4	3	2	1	0
\$3A	<b>INTF1</b>	<b>INTF0</b>	<b>INTF2</b>	-	-	-	-	-
L/E	L/E	L/E	L/E	L	L	L	L	L

Les cinq premiers bits (0 à 4) ne sont pas utilisés (lu à 0).

**INTF1, 0 External Interrupt Flag 1 ou 0** Quand un changement d'état apparaît sur la broche **INT1** (ou **INT0**) il déclenche une interruption et **INTF1** (ou **INTF0**) est mis à 1 si le bit dans **SREG** et le bit **INT1** (ou **INT0**) dans **GICR** est à un, le MCU sautera au vecteur d'interruption. Le drapeau est remis à 0 quand la routine d'interruption est exécutée. Le bit peut être remis à 0 par le programme avant la fin de la routine d'interruption. Ce drapeau est toujours remis à 0 quand **INT1** (ou **INT0**) est configuré en entrée d'interruption.

**INTF2 External Interrupt Flag 2** Quand un changement d'état sur la broche d'**INT2** déclenche une interruption et **INTF2** est mis à 1 si le bit dans **SREG** et le bit **INT2** dans **GICR** est à un, le MCU sautera au vecteur d'interruption. Le drapeau est remis à 0 quand la routine d'interruption est exécutée. Le bit peut être remis à 0 par programme. Ce drapeau est toujours remis à 0 quand **INT2** est configuré en entrée d'interruption. Notez qu'en entrant en mode sommeil, l'interruption avec l'**INT2** est mise hors de service.

### Registre TIFR (Timer/Counter Interrupt Flag)

Le registre **TIFR** indique l'état des interruptions internes des modules Timer et Comparateur. Les interruptions seront prises en charge si le bit correspondant dans le registre **TIMSK** à été activé (mis à 1). Ce registre sera revu dans les chapitres consacrés aux Timer.

Adresse	7	6	5	4	3	2	1	0
\$38	<b>OCT2</b>	<b>TOV2</b>	<b>ICF1</b>	<b>OCF1A</b>	<b>OCF1B</b>	<b>TOV1</b>	<b>OCF0</b>	<b>TOV0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCF2 Output Compare Flag 2** Le bit est mis à 1 quand une donnée est arrivée et comparée entre le Timer/Counter2 et le contenu du registre d'**OCR2**. Le bit **OCF2** est remis à 0 par le matériel en exécutant le vecteur de l'interruption. Vous pouvez aussi forcer **OCF2** à 0.

**TOV2 Timer/Counter2 Overflow Flag** Le bit est mis à 1 quand le Timer/Counter2 déborde. **TOV2** est mis à 0 par le matériel lors de l'exécution de l'interruption. Vous pouvez aussi forcer **TOV2** à 0. Dans le mode **PWM**, ce bit est mis à 1 quand le Timer/Counter2 arrive à la valeur \$00.

**ICF1 Timer/Counter1, Input Capture Flag** Ce bit est mis à 1 quand un événement de capture arrive sur la broche **ICP1**. Quand le registre de capture **ICR1** est utilisé, le bit est mis à 1 par l'arriver d'une valeur supérieure à **ICR1**. **ICF1** est automatiquement remis à 0 quand l'interruption est exécutée. Vous pouvez aussi forcer **ICF1** à 0.

**OCF1A Timer/Counter1, Output Compare A Match Flag** Ce bit est modifié après que le cycle d'horloge du minuteur est atteint la valeur correspondant au registre **TCNT1**. Notez qu'une comparaison forcée en sortie avec **FOC1A** ne mettra pas le bit à un. **OCF1A** est automatiquement remis à 0 lors de l'exécution de l'interruption. Vous pouvez aussi forcer **OCF1A** à 0.

**OCF1B Timer/Counter1, Output Compare B Match Flag** Ce bit est modifié après que le cycle d'horloge du minuteur est atteint la valeur du registre **TCNT1**. Notez qu'une comparaison forcée en sortie avec **FOC1B** ne mettra pas le bit à un. **OCF1B** est automatiquement remis à 0 quand l'interruption est exécutée. Vous pouvez aussi mettre le bit à 0.

**TOV1 Timer/Counter1, Overflow Flag** La mise à 1 de ce bit dépend du mode de fonctionnement défini pour le Timer/Compteur1. Dans le mode normal et **CTC**, le bit **TOV1** est mis à 1 quand le minuteur déborde. **TOV1** est automatiquement remis à 0 quand l'interruption est exécutée. Vous pouvez aussi mettre le bit à 0.

**OCF0 Output Compare Flag 0** Le bit est mis à 1 quand une arrivée est comparée entre le Timer/Counter0 et les données du registre **OCR0**. **OCF0** est remis à 0 par le matériel lors de l'exécution de l'interruption. Vous pouvez aussi mettre le bit à 0.

**TOV0 Timer/Counter0 Overflow Flag** Le bit est mis à 1 quand le Timer/Counter0 déborde. **TOV0** est automatiquement remis à 0 quand l'interruption est exécutée. Vous pouvez aussi mettre le bit à 0. Dans le mode **PWM**, ce bit est mis à 1 quand le Timer/Counter0 arrive à la valeur \$00.

### Registre TIMSK (Timer/Counter Interrupt Mask)

Le registre **TIMSK** définit individuellement le masque pour les modules Timer, Compteurs et Comparateurs. Le masque autorise les interruptions si il est à 1, si il est à 0 l'interruption ne sera pas prise en compte. Lors de l'interruption, le registre **TIFR** contiendra le bit du module qui à activer l'interruption et qu'il faudra tester pour connaître le modules en cause. Ce registre sera revu dans les chapitres consacrés aux Timer.

Adresse	7	6	5	4	3	2	1	0
\$39	<b>OCIE2</b>	<b>TOIE2</b>	<b>TICIE1</b>	<b>OCIE1A</b>	<b>OCIE1B</b>	<b>TOIE1</b>	<b>OCIE0</b>	<b>TOIE0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCIE2 Timer/Counter2 Output Compare Match Interrupt Enable** Quand le bit est à 1 l'interruption est validé pour le Timer/Counter2 et le bit **OCF2** est modifié dans le registre **TIFR**.

**TOIE2 Timer/Counter2 Overflow Interrupt Enable** Quand le bit est à 1 l'interruption est validé pour le Timer/Counter2 et le bit **TOV2** est modifié dans le registre **TIFR**.

**TICIE1 Timer/Counter1, Input Capture Interrupt Enable** Quand le bit est à 1 l'interruption est validé pour le Timer/Counter1 et le bit **ICF1** est modifié dans le registre **TIFR**.

- OCIE1A** *Timer/Counter1, Output Compare A Match Interrupt Enable* Quand le bit est à 1 l'interruption est validé pour le Timer/Counter1 et le bit **OCF1A** est modifié dans le registre **TIFR**.
- OCIE1B** *Timer/Counter1, Output Compare B Match Interrupt Enable* Quand le bit est à 1 l'interruption est validé pour le Timer/Counter1 et le bit **OCF1B** est modifié dans le registre **TIFR**.
- TOIE1** *Timer/Counter1, Overflow Interrupt Enable* Quand le bit est à 1 l'interruption est validé pour le Timer/Counter1 et le bit **TOV1** est modifié dans le registre **TIFR**.
- OCIE0** *Timer/Counter0 Output Compare Match Interrupt Enable* Quand le bit est à 1 l'interruption est validé pour le Timer/Counter0 et le bit **OCF0** est modifié dans le registre **TIFR**.
- TOIE0** *Timer/Counter0 Overflow Interrupt Enable* Quand le bit est à 1 l'interruption est validé pour le Timer/Counter0 et le bit **TOV0** est modifié dans le registre **TIFR**.

# La Mémoire EEPROM

L'EEPROM est une mémoire programmable est effaçable électriquement. La particularité de cette mémoire et de pouvoir garder les informations stockées longtemps même hors tension.

L'ATMEGA32 contient 1024 octets de données dans la mémoire EEPROM. Elle est organisée comme un espace de données séparé, dans lequel des octets simples peuvent être lus et écrits. L'EEPROM a une endurance d'au moins 100,000 cycles d'écriture/effacement.

## L'accès en Lecture/Ecriture dans l'EEPROM

L'espace mémoire de l'EEPROM est accessible par l'utilisation de registres spéciaux d'accès. Le temps d'accès en écriture dans l'EEPROM est donnée dans la table suivante :

Symbole	Nombre de Cycle d'oscillateur RC interne	Temps Typique de Programmation
Ecriture EEPROM	8448	8.5 ms

Le temps d'attente d'écriture de l'EEPROM doit être géré par le logiciel utilisateur qui doit détecter la fin de l'écriture par l'intermédiaire du registre EECR. Si le code d'utilisateur contient les instructions d'écriture dans l'EEPROM, quelques précautions doivent être prises. VCC doit rester très stable pendant la programmation de l'EEPROM donc, vérifier que l'alimentation de votre montage soit parfaite, utiliser un régulateur de type LM7805 qui assure une tension stable suffisante et n'oubliez pas de filtrer avec des capacités. Pour empêcher d'écrire dans l'EEPROM involontairement, la procédure doit être suivie à la lettre. Quand l'EEPROM est lu, l'UC est interrompue pour quatre cycles d'horloge avant que l'instruction suivante ne soit exécutée. Quand l'EEPROM est écrit, l'UC est interrompue pour deux cycles d'horloge avant que l'instruction suivante ne soit exécutée. Les trois registres utilisés par l'EEPROM sont décrits dans le chapitre suivant.

### Registre EEAR (*The EEPROM Address Register*)

Le registre d'adressage de l'EEPROM est divisé en deux parties pour les mémoires supérieures à 256 octets.

Adresse	7	6	5	4	3	2	1	0
\$1F	-	-	-	-	-	-	EEAR9	EEAR8
L/E	L	L	L	L	L	L	L/E	L/E
\$1E	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**EEARx EEPROM Adresse Register** Ce registre contient l'adresse de la case EEPROM à écrire ou à lire.

### Registre EEDR (*The EEPROM Data Register*)

Le registre de données de l'EEPROM qui est lu ou écrit dans la mémoire EEPROM.

Adresse	7	6	5	4	3	2	1	0
\$1E	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**EEDRx EEPROM Data Register** Ce registre stocke les données à écrire ou à lire.

## Registre EECR (*The EEPROM Control Register*)

Le registre de contrôle de l'EEPROM permet de définir le fonctionnement de celle-ci.

Adresse	7	6	5	4	3	2	1	0
\$1C					<b>EERIE</b>	<b>EEMWE</b>	<b>EEWE</b>	<b>EERE</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**EERIE EEPROM Ready Interrupt Enable** Validation de l'interruption **EE\_RDY**, se produisant quand une opération de lecture ou écriture est terminée.

**EEMWE EEPROM Master Write Enable** Autorisation d'écriture dans la mémoire (voir procédure).

**EEWE EEPROM Write Enable** Seconde autorisation d'écriture dans la mémoire (voir procédure).

**EERE EEPROM Read Enable** Demande de lecture de la mémoire **EEPROM**.

## Procédure de Lecture/Ecriture dans l'EEPROM

Bien que l'utilisation de l'EEPROM à l'aide des registres reste simple, une procédure particulière est effectuée avant toute demande d'écriture dans cette mémoire, dans le but de sécuriser au maximum le bon fonctionnement de celle-ci.

### Procédure d'écriture

Test du bit **EEWE** = 0 afin de savoir si une procédure d'écriture est en cours.

Ecrire l'adresse où l'on souhaite écrire la donnée dans les registres **EEAR**.

Ecrire la donnée dans le registre **EEDR**.

Mettre à 1 le bit **EEMWE**.

Mettre à 1 le bit **EEWE** dans un délai inférieur à quatre cycles d'horloge sous peine d'annuler l'écriture.

### Procédure de lecture

Test du bit **EEWE** = 0 afin de savoir si une procédure d'écriture est en cours.

Ecrire l'adresse où l'on souhaite accéder à la donnée dans les registres **EEAR**.

Mettre à 1 le bit **EERE**.

Lecture de la donnée dans le registre **EEDR**.

## Exemple de programmation

L'exemple qui suit permet d'écrire un octet à l'adresse contenue dans r18:r17 de l'EEPROM :

```
EEPROM_write:                                ; Etiquette d'attente
    sbic  EECR,EEWE                            ; Test si l'écriture est terminée
    rjmp  EEPROM_write                        ; Boucle si pas terminé
    out   EEARH,r18                            ; Déclare l'adresse (r18:r17)
    out   EEARL,r17
    out   EEDR,r16                            ; Ecrire la donnée (r16) dans le registre de donnée
    sbi   EECR,EEMWE                          ; Ecrire un 1 dans EEMWE
    sbi   EECR,EEWE                          ; Départ de l'écriture dans EEPROM avec EEWE
    ret   ; Fin de la procédure d'écriture
```

L'exemple qui suit permet de lire un octet à l'adresse contenue dans r18:r17 de l'EEPROM :

```
EEPROM_read:                                ; Etiquette d'attente
    Sbic  EECR,EEWE                            ; Test si l'écriture est terminée
    Rjmp  EEPROM_read                        ; Boucle si pas terminé
    Out   EEARH,r18                            ; Déclare l'adresse (r18:r17)
    Out   EEARL,r17
    Sbi   EECR,EERE                          ; Départ de la lecture dans EEPROM avec EERE
    in    r16,EEDR                            ; Lire la donnée dans le registre
    ret   ; Fin de la procédure de lecture
```

Après coupure électrique du montage, on peut vérifier à nouveau que l'octet inscrit en mémoire est toujours présent.

# Les Entrées/Sorties Numériques (PORTx)

Les microcontrôleurs ATMEL sont pourvus de ports d'entrées/sorties numériques pour communiquer avec l'extérieur. Ces ports sont multidirectionnels et configurable broche à broche soit en entrée, soit en sortie. D'autres modes sont aussi utilisables comme des entrées analogiques, des fonctions spéciales de comparaison, de communication synchrone, ... Mais pour le moment nous allons voir la fonction Numérique des ports.

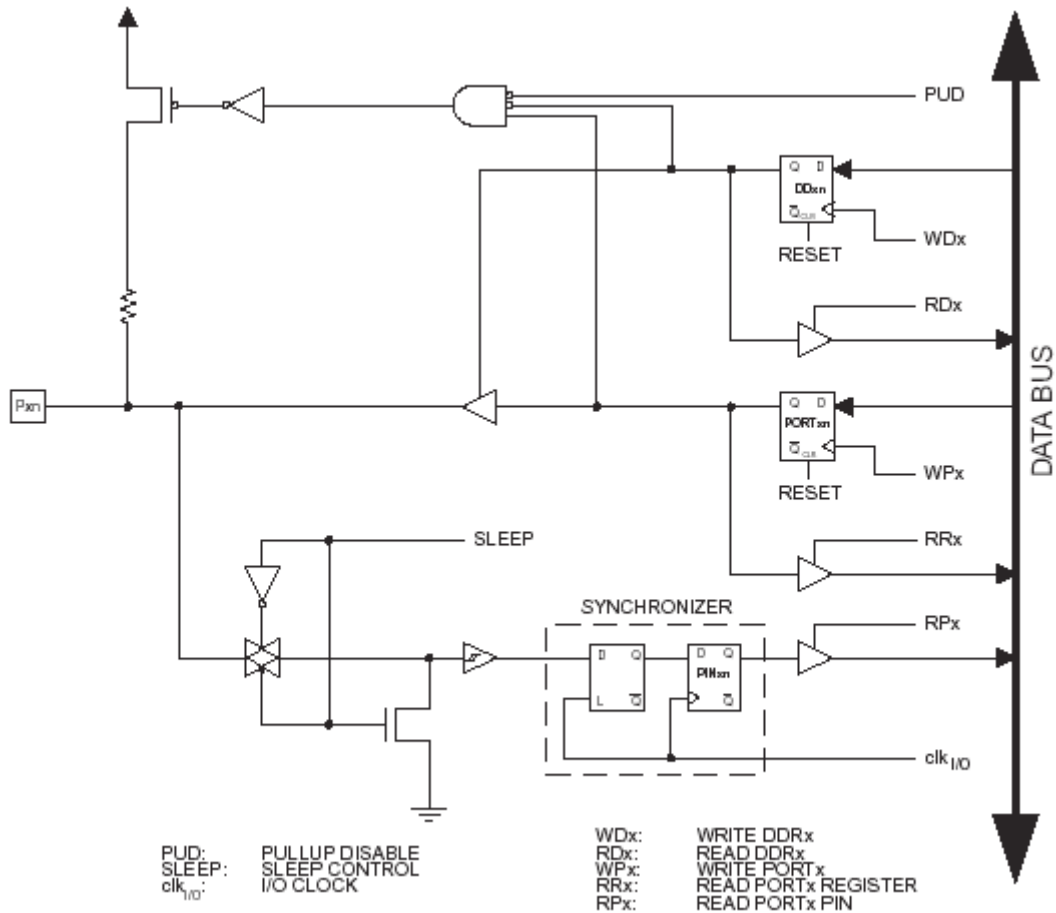


Figure 9, Synoptique des ports I/O.

## Généralité sur Les Ports

Pour éviter le flottement des valeurs en entrée d'un port, il est possible de mettre des résistances de tirage entre la broche du port et VCC (+3 ou +5 V), ce système est inclus dans le MCU, chaque port est configurable avec ou sans résistance de tirage.

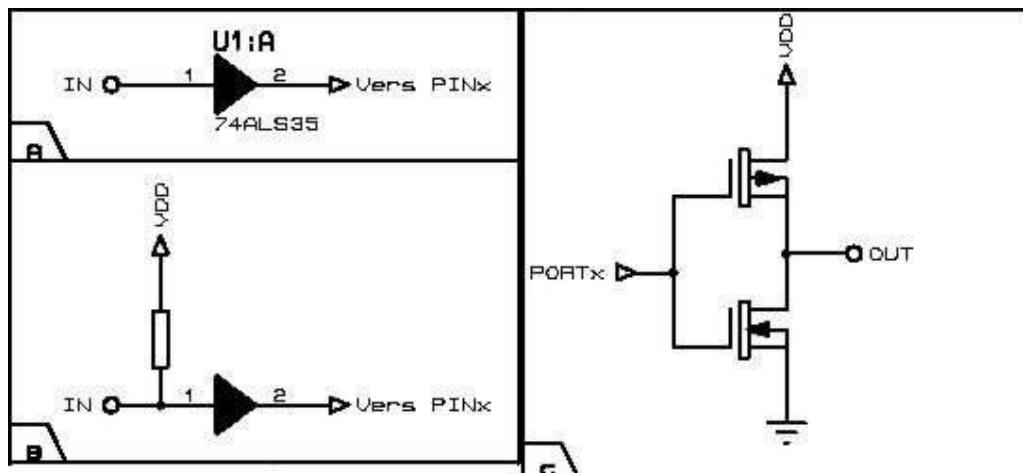


Figure 10, Schéma type des ports ATMEGA, Out en Push-Pull.

Le tableau suivant résume les configurations possibles :

DDRx	PORTx	PINx	Configuration
0	0	In	Entrée sans résistance de rappel (a)
0	1	In	Entrée avec résistance de rappel (b)
1	0	Out	Sortie à 0 (Zéro) (c)
1	1	Out	Sortie à 1 (un) (c)

Les entrées et sorties des ports sont schématisés en fonction des lettres entre parenthèses dans le figure 9.

La sortie Push-pull permet de délivrer un courant jusqu'à 40 mA, à utiliser en accord avec les 2 règles suivantes :

Chaque port de 8 bits est limité à un courant **total de 200 mA**,

Le microcontrôleur lui-même peut supporter au **maximum 400 mA**.

Donc, vous pouvez connecter 8 **LED** sur chaque port 8 bits et vous pouvez les allumer tous ensemble (8 x 20 mA = 160 mA), mais sur deux ports au maximum. La recommandation principale est de limiter le nombre de **LED** ou de mettre un amplificateur suiveur pour éviter de détériorer le microcontrôleur.

## Les Registres de Référence

Pour contrôler le mode de fonctionnement de chaque broche d'un port, 3 registres sont mis à notre disposition :

**DDRx** : Registre de direction (Mode Entrée, Sortie)

**PORTx** : Registre de données (Donnée à lire ou à écrire)

**PINx** : Registre d'état (Donnée disponible)

**x** Représente le nom de port (A, B, C ou D).

Chaque registre est configurable bit à bit, c'est à dire que sur l'on peu utiliser sur le même port des fonctions en entrée et/ou en sortie simultanément. Par exemple, on peut avoir les quatre premiers bits en entrée et les quatre derniers bits en sortie sur un même port (voir l'exemple de programmation).

### Registre DDRx (Data Direction Register)

Le registre **DDRx** permet de configurer le port en entrée avec la valeur 0 ou en sortie avec la valeur 1.

Adresse	7	6	5	4	3	2	1	0
\$1A	<b>DDRA7</b>	<b>DDRA6</b>	<b>DDRA5</b>	<b>DDRA4</b>	<b>DDRA3</b>	<b>DDRA2</b>	<b>DDRA1</b>	<b>DDRA0</b>
\$17	<b>DDRB7</b>	<b>DDRB6</b>	<b>DDRB5</b>	<b>DDRB4</b>	<b>DDRB3</b>	<b>DDRB2</b>	<b>DDRB1</b>	<b>DDRB0</b>
\$14	<b>DDRC7</b>	<b>DDRC6</b>	<b>DDRC5</b>	<b>DDRC4</b>	<b>DDRC3</b>	<b>DDRC2</b>	<b>DDRC1</b>	<b>DDRC0</b>
\$11	<b>DDRD7</b>	<b>DDRD6</b>	<b>DDRD5</b>	<b>DDRD4</b>	<b>DDRD3</b>	<b>DDRD2</b>	<b>DDRD1</b>	<b>DDRD0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**DDRx Data Direction Register** Le bit respectif d'un port peut être placé en entrée avec un 0 ou en sortie avec un 1.

### Registre PORTx (Data Register Port)

Le registre **PORTx** permet de configurer la résistance de tirage ou la valeur à sortir.

Adresse	7	6	5	4	3	2	1	0
\$1B	<b>PORTA7</b>	<b>PORTA6</b>	<b>PORTA5</b>	<b>PORTA4</b>	<b>PORTA3</b>	<b>PORTA2</b>	<b>PORTA1</b>	<b>PORTA0</b>
\$18	<b>PORTB7</b>	<b>PORTB6</b>	<b>PORTB5</b>	<b>PORTB4</b>	<b>PORTB3</b>	<b>PORTB2</b>	<b>PORTB1</b>	<b>PORTB0</b>
\$15	<b>PORTC7</b>	<b>PORTC6</b>	<b>PORTC5</b>	<b>PORTC4</b>	<b>PORTC3</b>	<b>PORTC2</b>	<b>PORTC1</b>	<b>PORTC0</b>
\$12	<b>PORTD7</b>	<b>PORTD6</b>	<b>PORTD5</b>	<b>PORTD4</b>	<b>PORTD3</b>	<b>PORTD2</b>	<b>PORTD1</b>	<b>PORTD0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**PORTx Data Register Port** Le bit respectif d'un port qui est en entrée sera connecté à une résistance de tirage si le bit est à 1 et si le bit **PUD** de **SFIOR** est à 0. Si le port est en sortie, le bit correspond à la valeur que l'on souhaite sur le port : 0 en bas, 1 en haut.

## Registre PINx (Reading the Pin Value)

Le registre **PINx** permet de lire le contenu d'un port quelle que soit sa configuration en entrée ou en sortie. Ce registre n'est accessible quant lecture, l'écriture ne sera considérée comme nul et sans effet.

Adresse	7	6	5	4	3	2	1	0
\$19	<b>PINA7</b>	<b>PINA6</b>	<b>PINA5</b>	<b>PINA4</b>	<b>PINA3</b>	<b>PINA2</b>	<b>PINA1</b>	<b>PINA0</b>
\$17	<b>PINB7</b>	<b>PINB6</b>	<b>PINB5</b>	<b>PINB4</b>	<b>PINB3</b>	<b>PINB2</b>	<b>PINB1</b>	<b>PINB0</b>
\$14	<b>PINC7</b>	<b>PINC6</b>	<b>PINC5</b>	<b>PINC4</b>	<b>PINC3</b>	<b>PINC2</b>	<b>PINC1</b>	<b>PINC0</b>
\$11	<b>PIND7</b>	<b>PIND6</b>	<b>PIND5</b>	<b>PIND4</b>	<b>PIND3</b>	<b>PIND2</b>	<b>PIND1</b>	<b>PIND0</b>
L/E	L	L	L	L	L	L	L	L

**PINx Reading the Pin Value** La valeur du bit respectif d'un port est lue quelque soit la configuration du port. Si le port est lu juste après une écriture sur **PORTx** ou **DDRx**, il faudra attendre la synchronisation d'un cycle d'horloge avec l'instruction **NOP** pour que la valeur soit correcte.

## Registre SFIOR (Special Function I/O Register)

Le registre spécial **SFIOR** permet de modifier le fonctionnement des entrées/sorties pour les ports (**PUD**).

Adresse	7	6	5	4	3	2	1	0
\$30	<b>ADTS2</b>	<b>ADTS1</b>	<b>ADTS0</b>	-	<b>ACME</b>	<b>PUD</b>	<b>PSR2</b>	<b>PSR10</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**PUD PUD: Pull-up disable** Quand le bit est à 1 les résistances de tirage sont désactivées et les ports en entrées sont en mode 3 états. Le bit à 0, les résistances de tirage sont activées.

## Programmation d'un Port Numérique

La programmation d'un port est relativement facile, il faut simplement définir l'utilisation préalable du port et y connecter l'électronique correspondante.

### Paramétrage d'un port

Pour définir le port en Entrée ou en sortie utilisée **DDRx**,

Si le port est en entrée, l'écriture d'un 0 ou d'un 1 positionnera la résistance de tirage dans **PORTx**,

Si le port est en sortie l'écriture d'un 0 ou d'un 1 positionnera le port dans **PORTx**,

La lecture de la donnée disponible sur le port est faite dans **PINx**,

Lorsque le port est en entrée, et que l'utilisation de résistance de tirage est effective un 0 doit être écrit dans **SFIOR**.

### Exemple de programmation

L'exemple qui suit permet de configurer le port B avec les 4 bits de poids faible en sortie avec les deux derniers bits 0&1 à l'état haut (1). Après initialisation du port, un temps d'attente est positionné pour la synchronisation et une lecture du port est réalisée. La valeur de la variable R16 sera positionnée en fonction des niveaux présents sur les 4 bits de poids fort et les deux derniers bits à 1 (le b devant les chiffres signifie Binaire) :

; Défini un port en sortie haute sur les broches 0 & 1 et défini la directions pour les broches du port B

```
ldi r16, b00000011 ; R16 deux bits en haut
```

```
ldi r17, b00001111 ; R17 4 bits en sortie
```

```
out PORTB,r16 ; Initialise le port B
```

```
out DDRB,r17 ; Initialise la direction
```

```
nop ; Pour la synchronisation
```

```
in r16,PINB ; Lecture du port B complet
```

; Ensuite vous pouvez traiter les bits de la variable R16, la valeur dans r16 sera bxxxx0011,x = selon le niveau.



# Le Comparateur Analogique (AC)

Le comparateur analogique compare les valeurs d'entrée sur la broche positive **AIN0 (PB2)** et la broche négative **AIN1 (PB3)**. Quand la tension sur la broche **AIN0** est plus haute que la tension sur la broche **AIN1**, le comparateur analogique **ACO** est mis à 1. Le comparateur analogique peut être utilisé pour déclencher la fonction de capture d'entrée du Timer/Compteur1. De plus, le comparateur analogique peut déclencher une interruption séparée exclusive. L'utilisateur peut choisir le front montant ou descendant pour déclencher la sortie **ACO**. L'entrée négative du comparateur peut être l'une des 8 entrées analogiques de l'ADC. On montre le synoptique du comparateur analogique dans la figure 10.

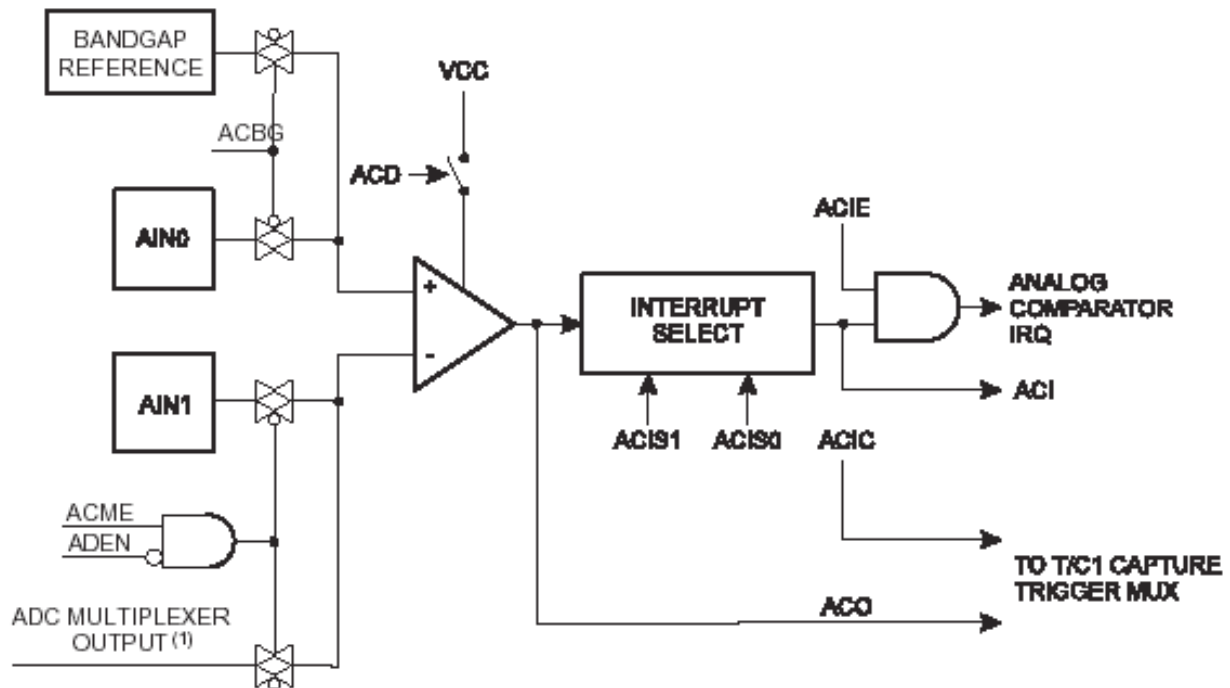


Figure 11, Synoptique du comparateur analogique.

## Entrée Multiplexé du Comparateur Analogique

Il est possible de choisir chacune des broches du convertisseur analogique/numérique **ADC7:0** pour remplacer l'entrée négative **AIN1** du comparateur analogique. Le multiplexeur de l'ADC (note 1 du synoptique) est employé pour choisir cette entrée et par conséquent l'ADC doit être éteint pour utiliser cette possibilité (**ADEN** dans **ADCSRA** est à 0).

Le choix entre l'entrée **AIN1** et l'entrée multiplexé de l'ADC est déterminé par le bit **ADEM** dans **SFIOR**. Les bits **MUX2:0** dans **ADMUX** choisissent la broche d'entrée ADC comme entrée négative du comparateur analogique, comme le montre le tableau suivant :

ACME	ADEN	MUX2:0	Entrée Négative du Comparateur Analogique
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

## Présentation des Registres du Comparateur Analogique

Le comparateur utilise un registre propre et partage deux registres avec d'autre module.

### Registre ACSR (*Analog Comparator Control and Status Register*)

Le registre de control **ACSR** défini le fonctionnement du comparateur analogique.

Adresse	7	6	5	4	3	2	1	0
\$08	<b>ACD</b>		<b>ACO</b>	<b>ACI</b>	<b>ACIE</b>	<b>ACIC</b>	<b>ACIS1</b>	<b>ACIS0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**ACD** *Analog Comparator Disable* Bit de mise en marche du comparateur analogique, la mise à 0 permet de le mettre en marche, la mise à 1 arrête le comparateur et réduit la consommation du microcontrôleur. Le comparateur peut être arrêté ou mis en marche à tout moment.

**ACO** *Analog Comparator Output* Contient le résultat de la comparaison : si **VAIN0** > **VAIN1** alors **ACO** = 1.

**ACI** *Analog Comparator Interrupt Flag* Bit de demande d'interruption quand la conditions sélectionnés est vérifiés (**ACIS1** et **ACIS0**). Ce bit est remis à 0 automatiquement après le traitement de l'interruption (si **ACIE** = 1).

**ACIE** *Analog Comparator Interrupt enable* Bit de validation de l'interruption **ANA\_COMP**.

**ACIC** *Analog Comparator Input Capture Enable* La mise à 1 de ce bit connecte la sortie du comparateur à l'entrée de capture du Timer1.

**ACIS1 & ACIS0** *Analog Comparator Interrupt Mode Select* Sélection du mode d'activation de la sortie **ACO** :

Mode d'activation de la Sortie ACO	<b>ACIS1</b>	<b>ACIS0</b>
Changement d'etat 1-->0 ou 0-->1	0	0
Réservé	0	1
Front descendant	1	0
Front montant	1	1

### Registre SFIOR (*Special Function I/O Register*)

Le registre spécial **SFIOR** permet de modifier le fonctionnement des entrées/sorties pour le comparateur analogique (**ACME**).

Adresse	7	6	5	4	3	2	1	0
\$30	<b>ADTS2</b>	<b>ADTS1</b>	<b>ADTS0</b>	-	<b>ACME</b>	<b>PUD</b>	<b>PSR2</b>	<b>PSR10</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**ACME** *Analog Comparator Multiplexer Enable* Quand ce bit est écrit à 1 et l'**ADC** est éteint (**ADEN** dans **ADCSRA** est à zéro), le multiplexeur **ADC** choisit l'entrée négative du comparateur analogique. Quand ce bit est à zéro **AIN1** est appliqué à l'entrée négative du comparateur analogique.

### Registre ADMUX (*ADC Multiplexer Selection Register*)

Seul la partie multiplexage est utilisée par le comparateur analogique dans ce registre.

Adresse	7	6	5	4	3	2	1	0
\$07	<b>REFS1</b>	<b>REFS0</b>	<b>ADLAR</b>	<b>MUX4</b>	<b>MUX3</b>	<b>MUX2</b>	<b>MUX1</b>	<b>MUX0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**MUX4, 3, 2, 1 & 0** *Analog Channel and Gain Selection Bits* Choix du canal **ADC** pour l'entrée négative du comparateur analogique en remplacement de **AIN1** :

<b>MUX4..0</b>	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
<b>0000</b>	ADC0	Non Applicable		
<b>0001</b>	ADC1			
<b>0010</b>	ADC2			
<b>0011</b>	ADC3			
<b>00100</b>	ADC4			

<b>00101</b>	ADC5	
<b>00110</b>	ADC6	
<b>00111</b>	ADC7	

## Programmation du Comparateur Analogique

Pour tester le comparateur analogique avec une **LED** et deux potentiomètres. Une tension mesuré sur un potentiomètre de 10 **Ko** connecter entre +5 V et la masse, la sortie varie donc de 0 à 5 V et sera connecter sur le broche **AIN0 (PB2)**. Le point de consigne est définie par un autre potentiomètre connecter de la même manière +5 V et masse qui est câblé sur **AIN1 (PB3)**. Nous aurons donc l'équation suivante :

Si **AIN0 > AIN1** alors Alarme = 1 (si tension > point de consigne alors comparaison = 1).

Si **AIN0 < AIN1** alors comparaison = 0.

### Paramétrage du comparateur Analogique

Autorisation d'interruption générale **SEI** = 1 si nécessaire,

Mise en marche du convertisseur **ACD** = 1,

Déclenchement de l'interruption par **ACIS1** = 0,

Changement d'état à la sortie de comparaison **ACIS0** = 0

Autorisation du comparateur à générer un interruption **ACIE** = 1

### Un exemple de programmation

```

ANA_COMP:  in    sauvreg,sreg           ; sauvegarde de SREG
           push  sauvreg              ; SREG est poussé dans la pile
           push  a                    ; a est poussé sur la pile
           in    a,acsr               ; chargement de ASCR dans a
           cbr   a,$DF                ; met à 0 a dans le but de récupérer la valeur du bit ACO
           cpi   a,$20                ; compare a avec 20H (si bit ACO = 1)
           brne  noalarme             ; si différent alors aller a noalarme (test du bit Z)
           sbi   portc,1              ; si égal alors led=1 (alarme=1)
           rjmp  anaend               ; passer la routine de noalarme
noalarme:  cbi   portc,1              ; éteindre la led d'alarme
anaend:    pop   a                    ; a est restitué
           pop   sauvreg              ; sauvreg est restitué
           out   sreg,sauvreg         ; sreg est remis dans son contexte (avant interruption)
           reti                       ; fin de l'interruption

;Programme de d'Initialisation
RESET:    ldi   a,low(RAMEND)
           out   SPL,a                ; Initialisation de la pile à
           ldi   a,high(RAMEND)       ; l'adresse haute de la SRAM
           out   SPH,a
           ldi   a,$00                ; port b en entrée
           out   ddrb,a
           ldi   r16,$00
           out   portb,a              ; sans résistance de rappel
           ldi   a,$FF                ; port c en sortie
           out   ddrc,a
           ldi   a,$00
           out   portc,a              ; sortie = 0
           ldi   a,$08                ; initialisation du comparateur
           out   acsr,a              ; avec interruption sur changement de front
           sei                               ; autorisation interruption general
debut:    cbi   portc,1              ; PC1 =0
           ...

```

# Le Convertisseur Analogique / Numérique (ADC)

Le convertisseur analogique/numérique **ADC** intégré dans l'**ATMEGA** est doté de caractéristique très intéressante avec une résolution sur 10 bits, 8 entrées simultanées avec une non linéarité inférieur à 1/2 **LSB**, une erreur à 0 V inférieur à 1 **LSB**, le temps de conversion est réglable de 65 à 260  $\mu\text{S}$  plus le temps est long, plus le résultat est précis. Près de 15000 échantillons/secondes avec le maximum de résolution son possible. Le convertisseur possède 7 entrées différentielles normales et 2 entrées différentielles avec gain optionnel de 10x (et 200x sur les boîtiers carré **TQFP** et **MLF** uniquement). La tension de référence peut être externe (conversion de 0 à **AREF** analogique, le maximum étant **VCC**) ou peut être interne avec la tension de référence de **2,56 V**. L'**ADC** à une interruption sur conversion complète. La limitation du bruit en mode de sommeil est possible.

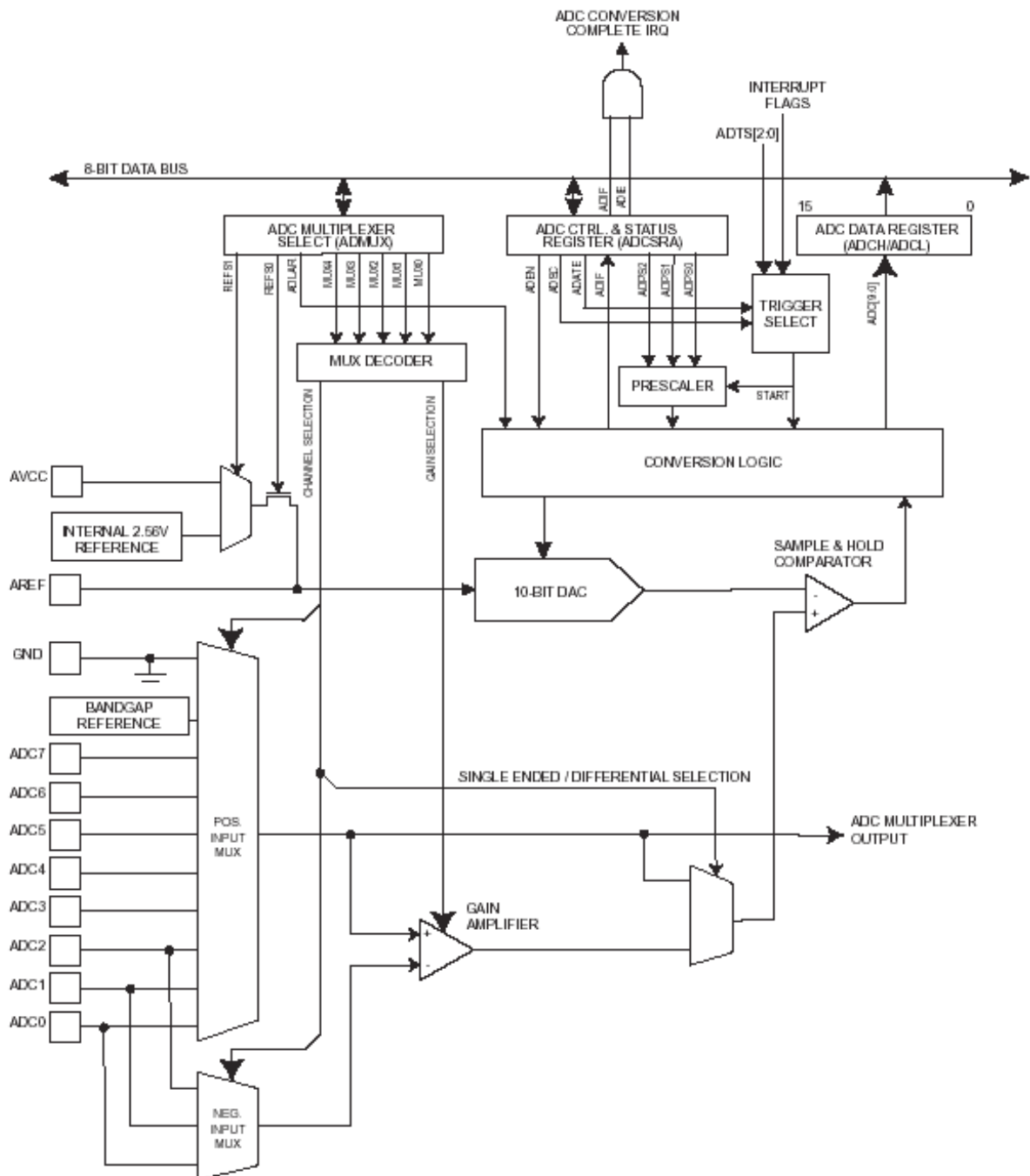


Figure 12, Synoptique du convertisseur AD.

## Fonctionnement de l'ADC

Le convertisseur étant chargé de convertir une tension analogique en résultat numérique (digital) codé sur 10 bits, nous pouvons écrire l'équation suivante :

$$\text{Résultat numérique} = (\text{Tension d'entrée} / \text{tension de référence } \mathbf{AREF}) \times 1024 - 1$$

Par exemple, pour avoir le résultat d'une tension d'entrée de 2,5 V avec une tension de référence de 5 V (**AREF**) nous aurons :

$$R_n = (2,5/5) \times 1024 = 511$$

Le temps de conversion est égal à 13 fois l'horloge système multiplié par le facteur de pré division. Afin de réduire au maximum les erreurs de conversions dues à la logique interne du contrôleur, des solutions peuvent être mises en oeuvre comme : Mettre en sommeil l'unité centrale avant le lancement d'une conversion, découpler soigneusement l'alimentation avec des condensateurs, respecter les règles élémentaires du routage de circuit imprimé en analogiques (connexions courtes, plan de masse...), dans tout les cas, effectuer toujours un traitement numérique des résultats (amortissement, moyenne, ...).

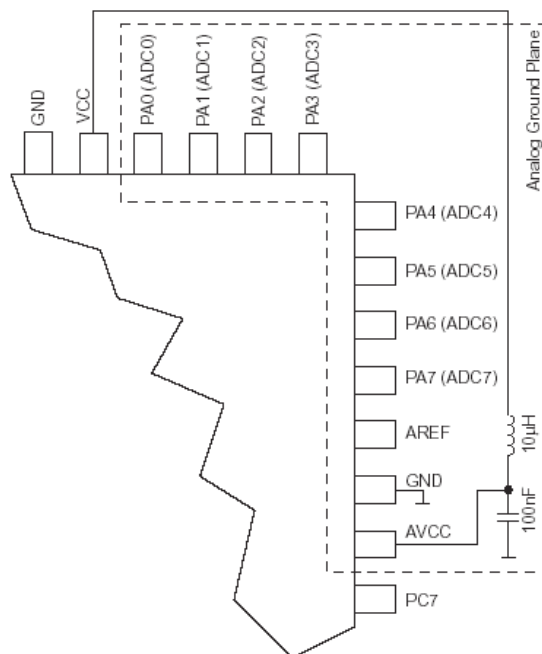


Figure 13, Plan de connexion du filtre d'alimentation.

L'ADC convertit une tension d'entrée analogique en une valeur à 10 bits digitale par approximation successive. La valeur minimale est **GND** et la valeur maximale est la tension sur la broche **AREF** moins 1 LSB. Facultativement la tension de référence **AREF** peut être connectée à **AVCC** ou une tension interne de 2,56 V en modifiant la valeur **REFSn** dans le Registre **ADMUX**. La référence de tension interne peut être découplée par un condensateur externe avec la broche **AREF** pour améliorer l'immunité aux bruits.

Les entrées analogiques à gain différentiel sont choisies en écrivant dans le bit **MUX** dans **ADMUX**. Chacune des 8 broches d'entrée ADC ainsi que **GND** et la référence de tension, peut être choisie comme des entrées simples de l'ADC. Les broches d'entrées ADC peuvent être choisies comme des entrées positives et négatives de l'amplificateur de gain différentiel. Si des canaux différentiels sont choisis, l'étage de gain différentiel amplifie la tension différence par le facteur de gain choisi. Cette valeur amplifiée devient alors le résultat de l'entrée analogique de l'ADC. Si des canaux simples sont employés l'amplificateur de gain est contourné.

L'ADC est actif avec le bit **ADEN** à 1 dans **ADCSRA**. La référence de tension et le choix du canal d'entrée n'entrera pas en vigueur quand **ADEN** est mis à 1, il faut d'abord désactiver **ADEN**. L'ADC ne consomme pas de puissance quand **ADEN** est à zéro, donc on recommande d'éteindre l'ADC avant l'entrée en mode sommeil.

L'ADC produit un résultat sur 10 bits qui est présenté dans les registres **ADCH** et **ADCL**. Par défaut, le résultat est présenté ajusté à droite, mais peut facultativement être présenté ajusté à gauche en mettant le bit

**ADLAR** dans **ADMUX**. Pour un résultat sur 8 bits et un ajustement à gauche, la lecture du registre **ADCH** est suffisante. Autrement, **ADCL** doit être lu en premier puis **ADCH** pour assurer la cohérence des données qui appartiennent à la même conversion. Une fois **ADCL** lu, l'accès aux registres de commandes est bloqué afin d'empêcher une nouvelle conversion tant que **ADCH** n'est pas lu. Quand **ADCH** est lu, l'ADC peut à nouveau être opérationnelle.

L'ADC a sa propre interruption qui peut être déclenché quand une conversion est achevée. Quand l'accès de l'ADC aux registres de commandes est interdit par la lecture d'**ADCL** et d'**ADCH**, l'interruption est quant même déclenché et le résultat sera perdu, il faut donc faire attention à lire rapidement le résultat d'une conversion.

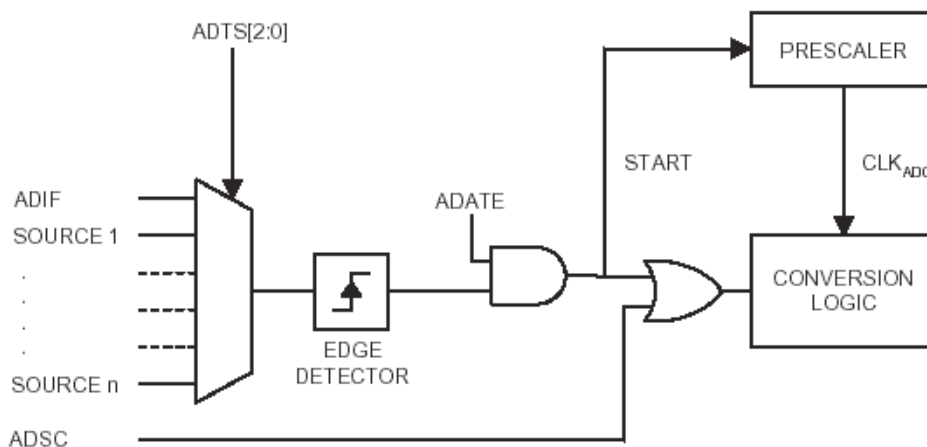


Figure 14, Synoptique de la logique de commande du convertisseur **AD**.

Une conversion simple est commencée en écrivant à 1 le bit de début de conversion **AD** dans **ADSC**. Le bit est maintenu en état haut tant que la conversion se réalise et sera remis à 0 par le matériel quand la conversion est achevée. Si un canal de données différent est choisi tandis qu'une conversion est en cours, l'ADC finira la conversion actuelle avant l'exécution du changement de canal.

Une conversion peut être déclenchée automatiquement par des sources diverses. Le déclenchement automatique qui met en marche l'ADC est programmable avec le bit **ADATE** dans **ADCSRA**. La source de déclenchement est choisie en mettant les bits de déclenchement **ADTS** dans **SFIOR** (voir ce registre dans les registres système). Quand un front positif arrive sur le signal de déclenchement choisi, le pré diviseur est remis à 0 et une conversion est commencée. Si le signal de déclenchement est toujours mis quand la conversion s'achève, une nouvelle conversion ne sera pas commencée. Si un autre front positif arrive sur le signal de déclenchement pendant la conversion, le front sera ignoré. Notez que le drapeau d'interruption sera mis même si l'interruption spécifique est mise hors de service ou l'interruption global **I** dans **SREG** est remis à 0. Une conversion peut ainsi être déclenchée sans causer une interruption. Cependant, le drapeau d'interruption doit être remis à 0 pour déclencher une nouvelle conversion et autorisé une interruption pour l'évènement suivant.

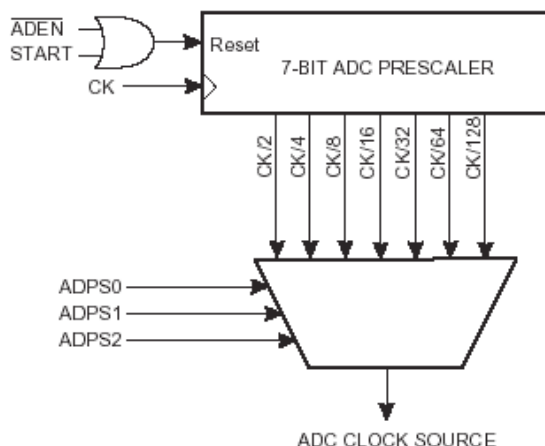


Figure 15, Synoptique du pré diviseur du convertisseur **AD**.

Par défaut, le circuit d'approximation successif exige une fréquence d'horloge d'entrée entre 50 **kHz** et 200 **kHz** pour obtenir la résolution maximale. Si une résolution inférieure à 10 bits est nécessaire, la fréquence d'horloge d'entrée de l'ADC peut être supérieure à 200 **kHz** pour augmenter le nombre de conversion à la seconde.

Le module **ADC** contient un pré diviseur qui produit une fréquence d'horloge acceptable pour l'ADC avec n'importe quelle fréquence d'UC de plus de 100 **KHz**. Le pré diviseur est mis en service par des bits dans **ADCSRA**. Le pré diviseur commence à compter dès que l'ADC est allumé en mettant le bit **ADEN** à 1 dans **ADCSRA**. Le pré diviseur continu à courir le bit **ADEN** est mis à 1.

Une conversion simple commence en mettant le bit **ADSC** à 1 dans **ADCSRA** puis en attendant le front montant suivant du cycle d'horloge **ADC**.

Une conversion normale prend 13 cycles d'horloge. La première conversion après l'allumage de l'ADC (**ADEN** mis à 1 dans **ADCSRA**) prend 25 cycles d'horloge pour initialiser le circuit analogique.

L'échantillon réel du signal a lieu 1,5 cycles d'horloge après le début d'une conversion normale et 13,5 cycles d'horloge après le début d'une première conversion. Quand une conversion est complète, le résultat est écrit dans le registre de données et **ADIF** est mis à 1. Dans le mode simple de conversion, **ADSC** est remis à 0 simultanément. Le logiciel peut alors lancer **ADSC** de nouveau et une nouvelle conversion sera amorcée sur le premier front montant de l'horloge.

Quand le déclenchement automatique est employé, le pré diviseur est remis à 0 quand l'événement de déclenchement arrive. Cela assure un retard pour le début de la conversion. Dans ce mode, l'échantillonnage a lieu 2 cycles d'horloge après le front montant du signal de déclenchement. Trois cycles d'horloge d'UC complémentaires sont employés pour la logique de synchronisation.

En employant le mode différentiel, avec déclenchement automatique d'une autre source, chaque conversion exigera 25 cycles d'horloge en raison de la mise hors de service et la réinitialisation après chaque conversion.

Dans le mode de course, une nouvelle conversion sera commencée immédiatement après que la conversion soit achevée, tandis qu'**ADSC** reste haut.

Condition	Echantillonnage (Cycles de Départ de Conversion)	Temps de Conversion (Cycles)
Première conversion	14.5	25
Normal, Simple	1.5	13
Auto Déclenchement	2	13.5
Normal, Différentielle	1.5/2.5	13/14

## Présentation des registres de l'ADC

Les registres utilisés par l'ADC son aux nombres de quatre.

### Registre ADMUX (ADC Multiplexer Selection Register)

Sélection de la voie de conversion (sur le port A, configuration des broches en entrées sans résistance de rappel).

Adresse	7	6	5	4	3	2	1	0
\$07	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**REFS1 & 0 Reference Selection Bits** Sélection des tensions de référence pour l'ADC, le tableau qui suit présente les choix possibles :

REFS1	REFS0	Sélection de la Tension de Référence
0	0	AREF Interne, VREF à l'arrêt
0	1	AVCC avec capacité externe sur AREF
1	0	Réservé
1	1	Tension de référence interne de 2,56 V avec capacité externe sur AREF

**ADLAR ADC Left Adjust Result** Ajustement à gauche à 1 ou à droite à 0 du résultat dans le registre **ADCL** et **ADCH**.

**MUX4, 3, 2, 1 & 0 Analog Channel and Gain Selection Bits** Choix du canal **ADC** et du Gain en fonction du tableau qui suit :

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain	
00000	ADC0	N/A			
00001	ADC1				
00010	ADC2				
00011	ADC3				
00100	ADC4				
00101	ADC5				
00110	ADC6				
00111	ADC7				
01000	N/A	ADC0	ADC0	10x	
01001		ADC1	ADC0	10x	
01010 <sup>(1)</sup>		ADC0	ADC0	200x	
01011 <sup>(1)</sup>		ADC1	ADC0	200x	
01100		ADC2	ADC2	10x	
01101		ADC3	ADC2	10x	
01110 <sup>(1)</sup>		ADC2	ADC2	200x	
01111 <sup>(1)</sup>		ADC3	ADC2	200x	
10000		ADC0	ADC1	1x	
10001		ADC1	ADC1	1x	
10010		ADC2	ADC1	1x	
10011		ADC3	ADC1	1x	
10100		ADC4	ADC1	1x	
10101		ADC5	ADC1	1x	
10110		ADC6	ADC1	1x	
10111		ADC7	ADC1	1x	
11000		ADC0	ADC2	1x	
11001		ADC1	ADC2	1x	
11010		ADC2	ADC2	1x	
11011		ADC3	ADC2	1x	
11100		ADC4	ADC2	1x	
11101		ADC5	ADC2	1x	
11110		1,22 V (V <sub>BG</sub> )	N/A		
11111		0 V (GND)			

(1) Les canaux d'entrée différentiels ne sont pas installés dans les boîtiers **PDIP**. Cette particularité est garantie seulement pour les boîtiers aux formats **TQFP** et **MLF**.



## Registre ADCSRA (ADC Control and Status Register A)

Le registre de contrôle et statut de l'ADC.

Adresse	7	6	5	4	3	2	1	0
\$06	<b>ADEN</b>	<b>ADSC</b>	<b>ADATE</b>	<b>ADIF</b>	<b>ADIE</b>	<b>ADPS2</b>	<b>ADPS1</b>	<b>ADPS0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**ADEN ADC Enable** Mise en marche du convertisseur avec la mise à 1 du bit, l'arrêt avec la mise à 0, la conversion en cours sera terminée.

**ADSC ADC Start Conversion** Lancement de la conversion de la voie sélectionnée (retourne à 0 en fin de conversion). En mode simple conversion il faut remettre à 1 à chaque nouvelle conversion. En mode libre, la première conversion dure 25 cycles puis les suivantes 15, il n'est pas nécessaire de remettre le bit à 1 à chaque conversion.

**ADATE ADC Auto Trigger Enable** La mise à 1 de ce bit permet de mettre en le convertisseur en fonction d'un déclencheur. La sélection du déclencheur est faite avec le bit **ADTS** du registre **SFIOR**.

**ADIF ADC Interrupt Flag** Passe à 1 une fois la conversion terminée et déclenche l'interruption si **ADIE** =1. Ce bit repasse automatiquement à 0 lors du traitement de la routine d'interruption.

**ADIE ADC Interrupt Enable** Validation de l'interruption **ADC**, déclenché lors du passage à 1 de **ADIF**.

**ADSP2...ADSP0 Prescaler Select Bits** Sélection du facteur de pré-division de l'horloge interne du convertisseur en fonction du quartz :

ADSP2	ADSP1	ADSP0	Facteur de division
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

## Registre ADCH et ADCL (ADC Data Register)

Registres de résultats de la conversion analogique / digital avec **ADLAR** = 0 :

Adresse	7	6	5	4	3	2	1	0
\$05	-	-	-	-	-	-	<b>ADC9</b>	<b>ADC8</b>
L/E	L	L	L	L	L	L	L/E	L/E
\$04	<b>ADC7</b>	<b>ADC6</b>	<b>ADC5</b>	<b>ADC4</b>	<b>ADC3</b>	<b>ADC2</b>	<b>ADC1</b>	<b>ADC0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

Registres de résultats de la conversion analogique / digital avec **ADLAR** = 1 :

Adresse	7	6	5	4	3	2	1	0
\$05	<b>ADC9</b>	<b>ADC8</b>	<b>ADC7</b>	<b>ADC6</b>	<b>ADC5</b>	<b>ADC4</b>	<b>ADC3</b>	<b>ADC2</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
\$04	<b>ADC1</b>	<b>ADC0</b>						
L/E	L/E	L/E	L	L	L	L	L	L

## Registre SFIOR (*Special Function I/O Register*)

Le registre spécial **SFIOR** permet de modifier le déclenchement du convertisseur A/D (**ADTSx**).

Adresse	7	6	5	4	3	2	1	0
\$30	<b>ADTS2</b>	<b>ADTS1</b>	<b>ADTS0</b>	-	<b>ACME</b>	<b>PUD</b>	<b>PSR2</b>	<b>PSR10</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**ADTS2 :0 ADC Auto Trigger Source** Si **ADATE** dans **ADCSRA** est écrit à un, la valeur de ce bit détermine la source de déclenchement d'une conversion **ADC**. Si **ADATE** est mis à zéro, **ADTS2:0** n'auront aucun effet. Une conversion sera déclenchée par le bord montant de la source choisi. La commutation d'une source qui est mis à zéro produira un bord montant sur la source. Si **ADEN** dans **ADCSRA** est mis à 1, la conversion commencera. Si **ADTS2:0 = 0** il n'y a pas de déclenchement.

ADTS2	ADTS1	ADTS0	Source de déclenchement
0	0	0	Pas de déclenchement
0	0	1	Comparateur Analogique
0	1	0	Attente Interruption '0' Externe
0	1	1	Comparaison Timer/Compteur 0
1	0	0	Débordement Timer/Compteur 0
1	0	1	Comparaison Timer/Compteur 1
1	1	0	Débordement Timer/Compteur 1
1	1	1	Capture sur Timer/Compteur 1

## Programmation de l'ADC

### Paramétrage du convertisseur simple

Mise en marche du convertisseur **ADEN = 1**  
 Sélection du facteur de pré division **ADPS2...ADPS0**  
 Sélection de l'entrée à convertir **MUX4...MUX0**  
 Enclenchement d'une acquisition **ADSC = 1**  
 Attente fin de conversion **ADSC = 0**  
 Lecture du résultat sur **ADCH** et **ADCL**.

### Exemple de programmation

```

RESET:      ldi      a,low(RAMEND)    ; Initialisation de la pile à ...
            out      SPL,a
            ldi      a,high(RAMEND) ; l'adresse haute de la SRAM
            out      SPH,a
            ldi      a,$00        ; port a en entrée
            out      ddra,a
            ldi      r16,$00
            out      porta,a      ; sans résistance de rappel
            ldi      a,$FF        ; port b en sortie
            out      ddrb,a
            ldi      r16,$00
            out      portb,a      ; Mise à 0 du port B
            ldi      a,$FF        ; port c en sortie
            out      ddrc,a
            ldi      r16,$00
            out      portc,a      ; Mise à 0 du port c
            ldi      a,$87        ; mis en marche convertisseur analogique
            out      adcsr,a      ; mode mono échantillonnage (ADFR =0)
            rjmp     debut        ; avec facteur de pré division /128
  
```

;Programme Principal

```
debut:      ldi      a,$00
            out      admux,a      ; sélection de la voie 0
debu1:      sbi      adcsr,adsc   ; démarre la conversion
wait1:      sbic     adcsr,adsc   ; Test fin de conversion
            rjmp     wait1
            in       a, adcl      ; transfert de ADCL dans A
            out      portb,a     ; transfert de A dans PortB
            in       a, adch      ; transfert de ADCH dans A
            out      portc,a     ; transfert de A dans PortC
            rjmp     debut       ; Bouclage infini.
```

# Le Pré Diviseur des Timer/Compteur0 & 1

Le Timer/Compteur0 et le Timer/Compteur1 partagent le même module de pré division, mais les Timer/Compteurs peuvent avoir des options différentes de pré diviseur.

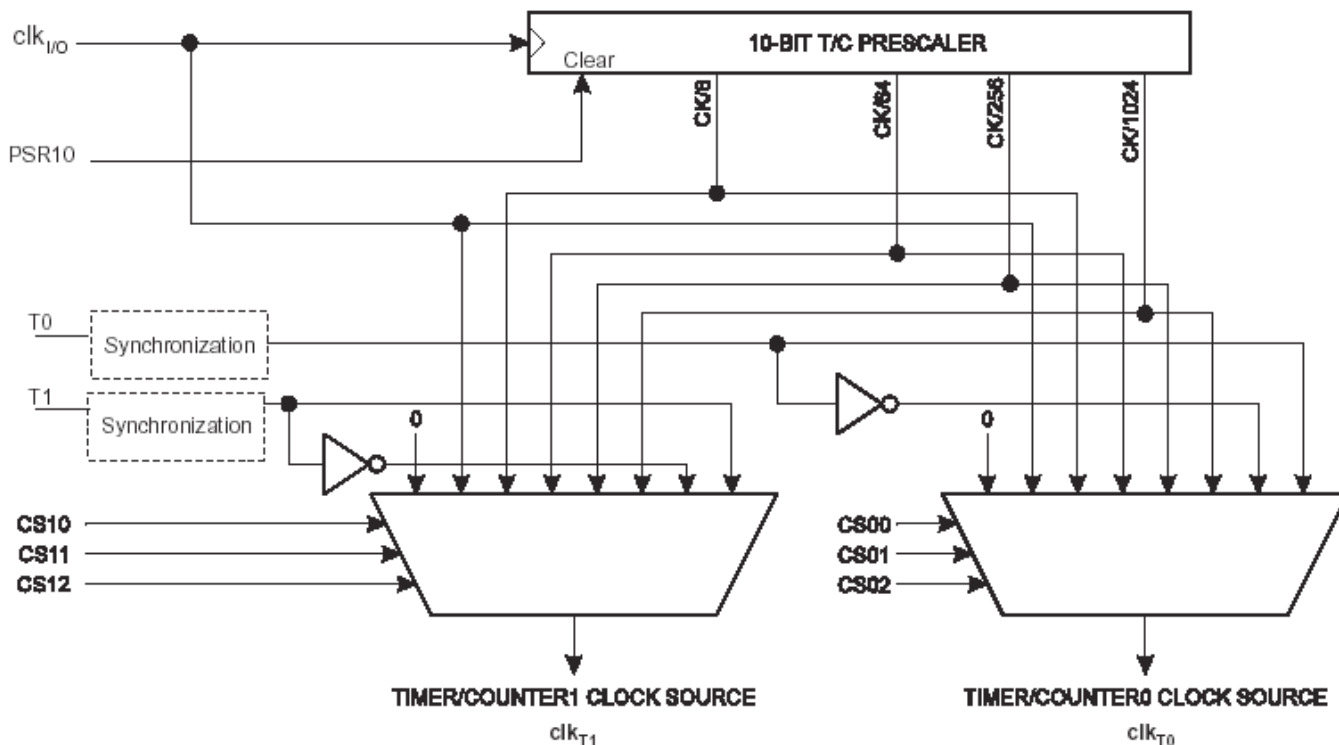


Figure 16, Synoptique du pré diviseur.

## Source d'Horloge Interne

Le Timer/Compteur peut être cadencé directement par l'horloge de système (en mettant le **CSN2:0** à 1) permettant une fréquence d'horloge maximal (**Fclk\_I/O**). Autrement, l'un des quatre bits du pré diviseur peut être employé comme la source d'horloge avec un facteur de pré division équivalent à : **Fclk\_I/O/8**, **Fclk\_I/O /64**, **Fclk\_I/O/256**, ou bien **Fclk\_I/O/1024**.

## Remise à 0 du pré diviseur

Le pré diviseur fonctionne indépendamment de l'horloge choisie par le Timer/Compteur et il est partagé par Timer/Compteur0 et Timer/Compteur1. Puisque le pré diviseur n'est pas affecté par la configuration de l'horloge du Timer/Compteur, l'état du pré diviseur aura des implications pour des situations où une horloge pré divisée est employée. Le nombre de cycles d'horloge du système sera pour le premier Timer/Compteur de 1 jusqu'à N+1, où N = la valeur du pré diviseur (1, 8, 64, 256, ou 1024). Il est possible d'employer la remise à 0 du pré diviseur pour synchroniser le Timer/Compteur sur l'exécution du programme. Cependant, un grand soin doit être pris si l'autre Timer/Compteur emploie aussi le pré diviseur. Un pré diviseur remis à 0 affectera la période de pré division pour tous Timer/Compteurs auxquels il est connecté.

## Source d'Horloge Externe

Une source d'horloge externe appliquée à la broche **T0/T1** peut être employée comme horloge de Timer/Compteur (**clkT1/clkT0**). Le signal sur **T0/T1** est échantillonné une fois par cycle d'horloge système par la logique de synchronisation. On passe alors le signal synchronisé par le détecteur de front. Les registres sont cadencés sur des fronts montant de l'horloge système interne (**clkI/O**) et reste transparent pour l'horloge système interne.

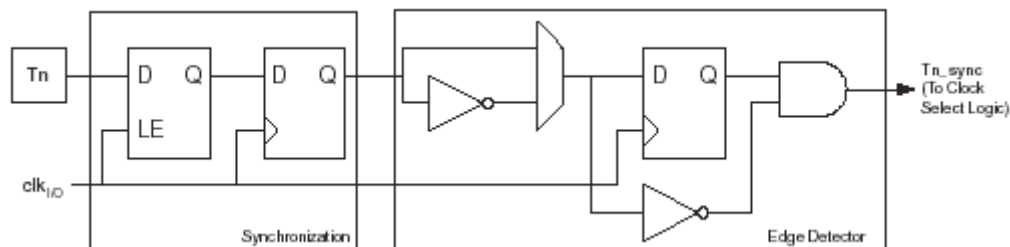


Figure 17, Echantillonnage du signale d'horloge externe.

Le détecteur de bord produit une impulsion **clkT0/clkT1** pour chaque front positif (**CSn2:0** à 7) ou négatif (**CSn2:0** à 6) de l'entrée qu'il détecte.

Le module de synchronisation et le détecteur de front présente un retard de 2,5 à 3,5 cycles d'horloge système entre l'entrée du signal appliqué à la broche **T0/T1** et l'arrivé au Timer/Compteur.

La demi période d'horloge externe doit être plus longue qu'un cycle d'horloge système pour assurer un bon échantillonnage ( $f_{ExtClk} < f_{clk\_I/O}/2$ ). Cependant, en raison de la variation possible de la fréquence d'horloge système, on recommande que la fréquence maximale d'une source d'horloge externe soit de moins de  $f_{clk\_I/O}/2,5$ . Une source d'horloge externe ne peut pas être pré divisé.

### Registre SFIOR (*Special Function I/O Register*)

Le registre spécial **SFIOR** permet de modifier le fonctionnement des entrées/sorties pour les Timer/Compteurs (**PSR10**).

Adresse	7	6	5	4	3	2	1	0
\$30	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	<b>PSR10</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**PSR10 Prescaler Reset Timer/Counter0 and Timer/Counter1** Quand ce bit est à un le pré diviseur du Timer/Counter0 et du Timer/Counter1 sera remis à 0. Le bit est remis à 0 par le matériel après l'exécution de l'opération. L'écriture d'un zéro dans ce bit n'aura aucun effet. Notez que le Timer/Counter0 et le Timer/Counter1 partagent même pré diviseur et qu'ils seront affecter tous les deux par ce bit. Ce bit sera toujours lu à zéro.

# Le Timer/Compteur0 à 8 Bits (TIMER0)

Un Timer/Compteur est un ensemble logique qui permet d'effectuer du comptage : de temps, d'évènements, de base de temps pour la génération de signaux...

Le Timer/Compteur0 est un compteur à 8 bits avec les particularités principales :

- Remise à 0 du compteur sur Comparaison (Rechargement Automatique)
- Générateur de Modulation de Phase en Largeur d'Impulsion (**PWM**)
- Générateur de Fréquence
- Compteur d'Évènement Externe
- Horloge à 9 bits avec pré diviseur
- Interruption sur Débordement et Comparaison (**TOV0** et **OCF0**).

Voici la vue le synoptique simplifié du Timers/Compteur à 8 bits.

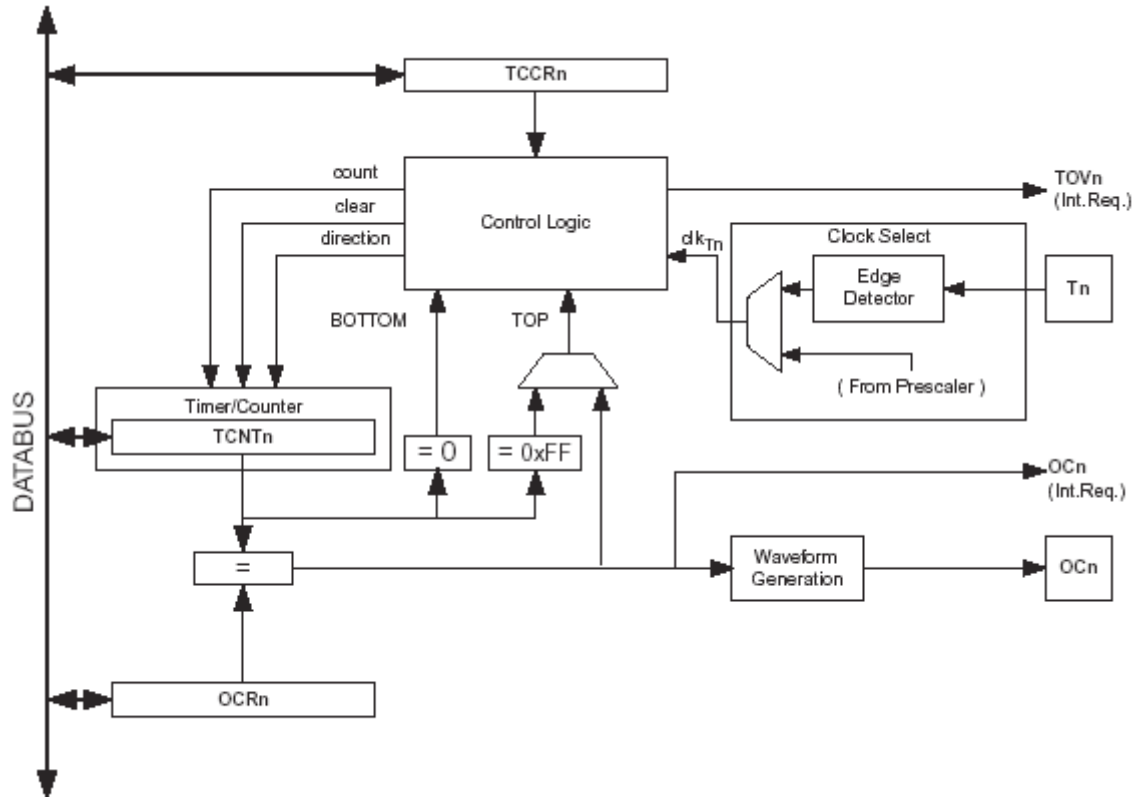


Figure 18, Synoptique du Timer/Compteur0.

## Fonctionnement du Timer/Compteur0

Le fonctionnement du Timer/Compteur0 repose sur un compteur **TCNT0** à 8 bits incrémenté et comparé en permanence au registre **OCR0** qui peut générer des actions sur la sortie **OC0** (**PB3**). Le choix de la source d'horloge **clkT0** peut être l'horloge interne, via le pré diviseur, une horloge externe sur la broche **T0** (**PB0**) ou inactif quand aucune source d'horloge n'est choisie. Le registre de commande **TCCR0** permet de choisir par mis les 4 modes de fonctionnement. La demande d'interruption (abrégé Int.Req. dans la figure) est visible dans le registre **TIFR**. Toutes les interruptions sont individuellement masquées dans **TIMSK** (Voir le chapitre sur les interruptions).

Le Générateur de Forme d'onde emploie les bits **COM01:0** pour définir l'état d'**OC0**. Dans la figure qui suit, le registre de contrôle du port d'entrée-sortie **DDR** et **PORT** est affecté par les bits **COM01:0**. Lors d'un **RESET** la sortie **OC0** est mise à 0.

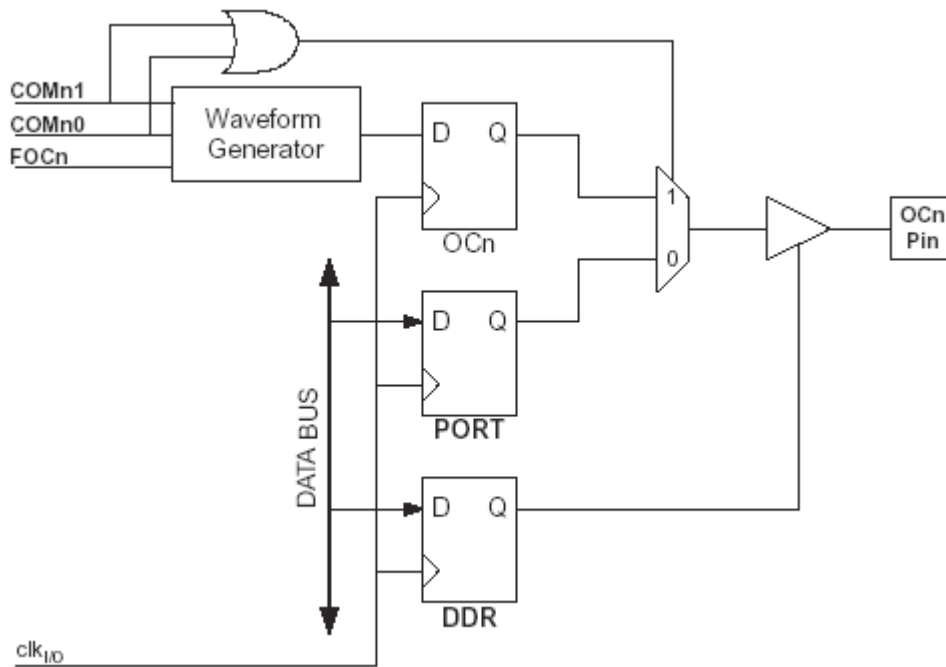


Figure 19, Synoptique de la mise en forme d'onde sur **OC0**.

### Le mode Normal du Timer/Compteur0

Le mode normal est le mode de fonctionnement le plus simple (**WGM01:0** à 0). Dans ce mode le comptage est toujours incrémental sans remise à 0 manuel du compteur. Il passe par le maximal **\$FF** en débordant, le drapeau **TOV0** est mis à 1, puis le compteur repart au début **\$00**. Le drapeau **TOV0** se comporte comme un neuvième bit mais il n'est pas remis à 0. Cependant, combiné avec l'interruption de débordement du Timer, le drapeau **TOV0** peut être remis à 0 automatiquement, augmentant d'un bit la résolution du Timer. Une nouvelle valeur peut être écrite n'importe quand dans le registre **TCNT0**.

Le Timer combiné avec l'unité de comparaison **OCR0** peut être employée pour produire une interruption au bout d'un temps donnée. L'utilisation de cette possibilité n'est pas recommandée pour la production de forme d'onde, puisque cela occupera trop de temps **CPU**, le mode suivant est préférable pour cela.

### Le mode Timer à Remise à 0 sur Comparaison (CTC)

Le mode **CTC** (**WGM01:0** à 2), le registre **OCR0** est employé pour définir la résolution du compteur. Le compteur est incrémentée et quant la valeur est égale à **OCR0**, le compteur est remis à **\$00** et le drapeau **OCF0** est mis à 1 pour déclencher une éventuelle interruption. **OCR0** définit la valeur supérieure pour le compteur donc sa résolution.

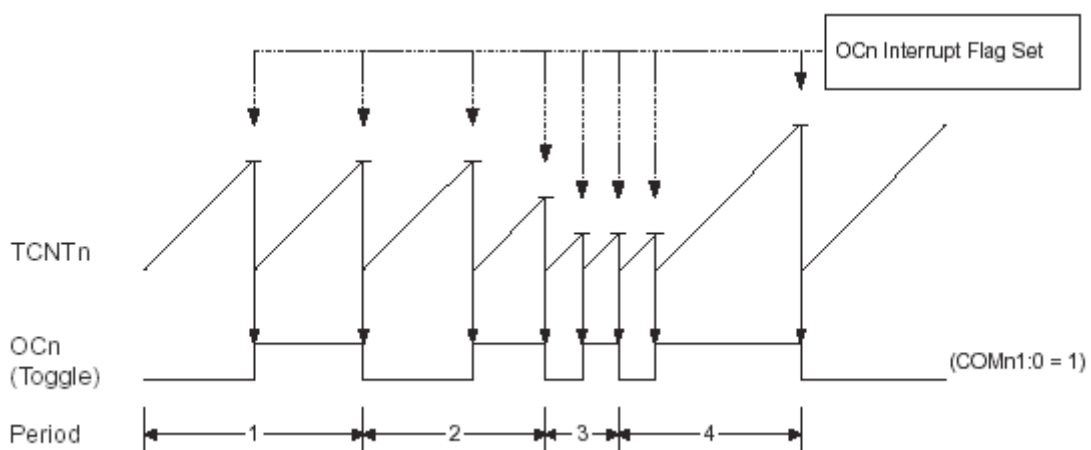


Figure 20, Mode **CTC** à remise à 0.

Le drapeau **OCF0** est automatiquement remis à 0 lors de l'exécution de l'interruption. Ce mode permet de mieux contrôler la fréquence produite. Il simplifie aussi l'opération de comptage des événements externes. La fréquence maximum que peut produire le Timer0 est fonction de la formule suivante :

$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

Avec **N** le rapport du pré diviseur (1, 8, 64, 256 ou 1024) et **OCR0 = \$00** pour le maximum, **Fclk\_I/O** étant la fréquence d'horloge du quartz.

### Le mode Modulation en Largeur d'Impulsion Rapide (PWM R)

Le mode **PWM rapide** permet de produire des impulsions de largeur variable en fonction des informations du registre **WGM01:0** à 3. Le compteur est incrémenté de **\$00** à **\$FF** puis reprend à **\$00**. Le bit **OC0 (PB3)** est mis à 1 au début du comptage puis il est mis à 0 lors de la comparaison entre **TCNT0** et **OCR0**. La remise à zéro du compteur **TCNT0** doit être faite dans la gestion d'interruption pour une période plus courte. Il est possible d'inverser le niveau **OC0** par les bits **COM1:0**. La forme d'onde n'est pas symétrique.

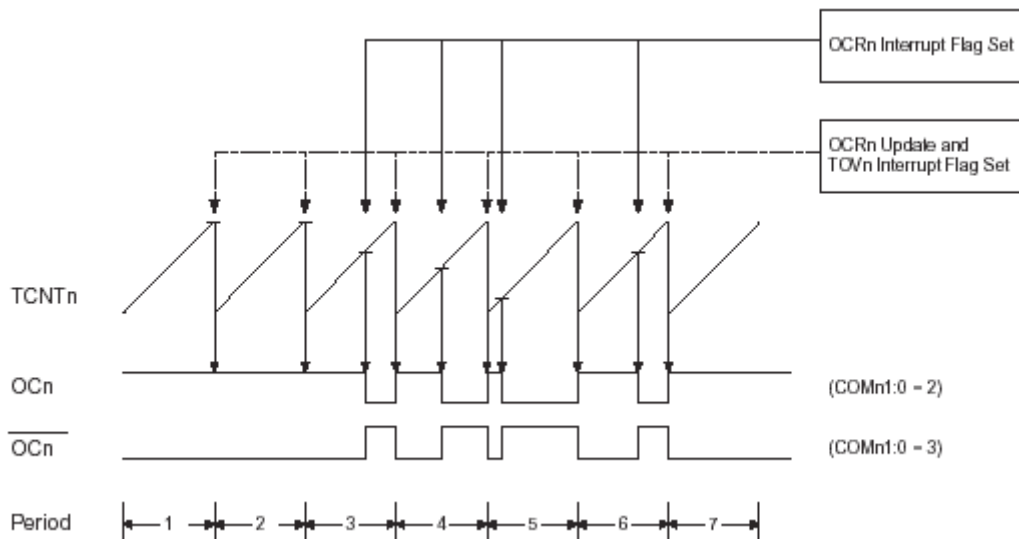


Figure 21, Mode PWM Rapide sur simple pente montante.

La fréquence maximum que peut produire le **PWM rapide** est fonction de la formule suivante :

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

Avec **N** le rapport du pré diviseur (1, 8, 64, 256 ou 1024), **Fclk\_I/O** étant la fréquence d'horloge du quartz.

### Le mode PWM Correct (PWM C)

Le mode **PWM correct** utilise le compteur dans les deux sens, quant la valeur définie dans le registre **OCR0** est atteinte la sortie **OC0** est changé et le compteur repart à l'envers. Quant il arrive à **\$00**, la valeur d'**OC0** est à nouveau changée. Le mode **PWM correct (WGM01:0** à 1) fournit une phase de haute résolution de forme d'onde **PWM**. Le **PWM correct** est basée sur une opération à double pente. Le compteur change de direction, il passe de **\$00** à **Max** et ensuite de **Max** à **\$00**. Dans le mode montant la sortie **OC0** est remise à 0 sur comparaison entre **TCNT0** et **OCR0** et dans le mode descendant, la sortie sur **OC0** est mise à 1 de même sur le cycle de monté descente ou les valeurs sont inversées.

Dans la phase corrigée, le compteur est incrémenté jusqu'à la valeur maximale avant d'être inversé et le compteur **TCNT0** sera égal au **Max** pour un cycle d'horloge complet du Timer2. Les petites marques horizontales sur les pentes de **TCNT0** représentent les comparaisons entre **OCR0** et **TCNT0**. Le drapeau de débordement **TOV0** est mis à 1 à chaque fois que le compteur atteint **\$00**.



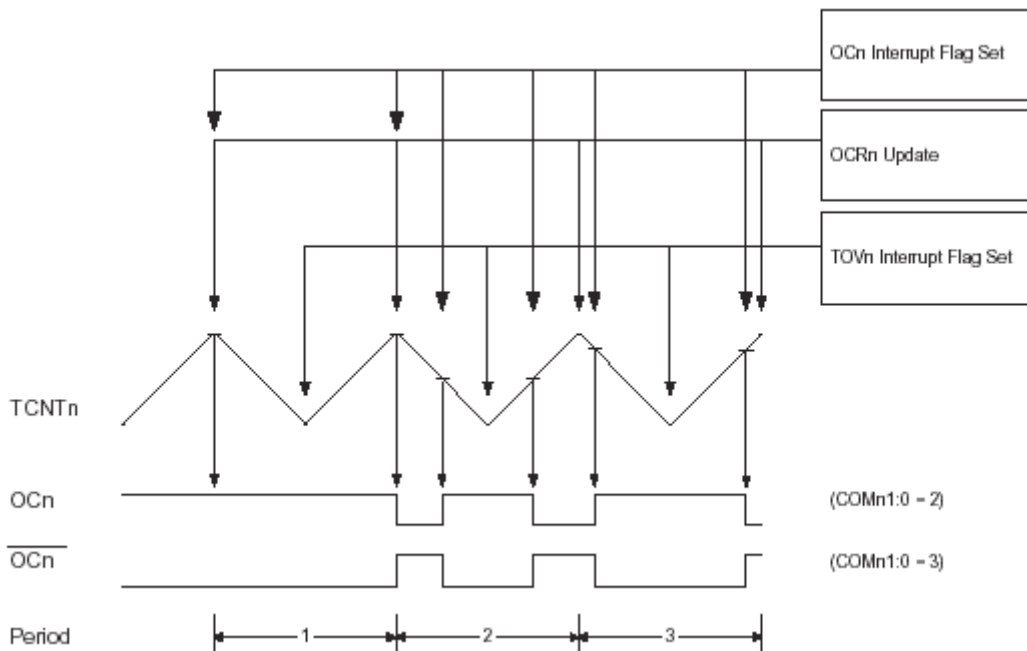


Figure 22, Mode **PWM Correct** sur double pente.

La fréquence est plus basse avec ce type de **PWM** mais présente l'avantage d'avoir un signal plus propre, préféré pour des applications de contrôle du moteur. La valeur de **TCNT0** sera égale à deux fois Max pour un cycle d'horloge de Timer. La fréquence maximum que peut produire le **PWM correct** est fonction de la formule suivante :

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

Avec **N** le rapport du pré diviseur (1, 8, 64, 256 ou 1024), **Fclk\_I/O** étant la fréquence d'horloge du quartz.

## Les Registres Associés au Timer/Compteur0

Le Timer0 utilise 3 registres spécifiques et deux registres d'interruption.

### Registre TCCR0 (Timer/Counter Control Register)

Registre de contrôle du Timer 0, sélection du facteur de pré division de l'horloge de comptage ou incrémentation par signaux externe sur broche **T0**.

Adresse	7	6	5	4	3	2	1	0
\$33	<b>FOC0</b>	<b>WGM00</b>	<b>COM01</b>	<b>COM00</b>	<b>WGM01</b>	<b>CS02</b>	<b>CS01</b>	<b>CS00</b>
L/E	E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**FOC0 Force Output Compare** Le bit **FOC0** est seulement actif quand le bit **WGM00** spécifie un mode non **PWM**. Cependant, pour assurer la compatibilité avec des dispositifs futurs, ce bit doit être mis à 0 quand **TCCR0** est écrit lors du fonctionnement en mode **PWM**. Le bit **FOC0** mis à 1 lance la comparaison immédiate et la sortie **OC0** est changée selon les bits **COM01:0**. Notez que le bit **FOC0** est mis en oeuvre comme un **strobe**. Le **FOC0 strobe** ne produira pas d'interruption et ne remettra pas à 0 le Timer dans le mode **CTC** employant **OCR0** comme valeur maximum. Le bit **FOC0** est toujours lu à zéro.

**WGM01:0 Waveform Generation Mode** Ces bits contrôlent l'ordre de comptage du Timer, la source de la valeur maximale (supérieure) et le genre de génération de forme d'onde à employer :

Mode	<b>WGM01 (CTC0)</b>	<b>WGM00 (PWM0)</b>	Timer/Counter	Bas à	Mise à jour <b>OCR0</b>	<b>TOV0 sur</b>
0	0	0	<b>Normal</b>	\$FF	Immédiat	Max
1	0	1	<b>PWM, Phase Correct</b>	\$FF	Bas (0)	Haut (255)
2	1	0	<b>CTC</b>	<b>OCR0</b>	Immédiat	Max
3	1	1	<b>PWM Rapide</b>	\$FF	Bas (0)	Max

**COM01:0 Compare Match Output Mode** Ces bits contrôlent la sortie **OC0**. Si les deux bits sont mis à 0, le port reprend son fonctionnement normal d'entrée-sortie. Cependant, notez que le registre de direction des données **DDR** dont correspondant à la broche **OC0** doit être positionnée en sortie pour permettre la production de signal **PWM**. Quand **OC0** est en service, la fonction des bits **COM01:0** dépend de la configuration de **WGM01:0** selon les trois tableaux qui suivent :

Mode de sortie de comparaison non **PWM**

COM01	COM00	Sortie sur OC0
0	0	Normal, déconnecté
0	1	Bascule sur comparaison
1	0	Mis à 0 sur comparaison
1	1	Mis à 1 sur comparaison

Mode de sortie de comparaison avec **PWM** rapide

COM01	COM00	Sortie sur OC0
0	0	Normal, déconnecté
0	1	Réservé
1	0	Remis à 0 sur comparaison, mis à 1 en Bas
1	1	Mis à 1 sur comparaison, mis à 0 en Bas

Mode de sortie de comparaison avec **PWM** correct

COM01	COM00	Sortie sur OC0
0	0	Normal, déconnecté
0	1	Réservé
1	0	Mis à 0 sur comparaison montante, mis à 1 sur comparaison descendante
1	1	Mis à 1 sur comparaison montante, mis à 0 sur comparaison descendante

**Note :** Un cas spécial arrive quand **OCR0** est égale au maximum et **COM01** est mis à 1. Dans ce cas, la comparaison est ignorée.

**CS02:0 Clock Select** Les trois bits de choix de l'horloge permettent de sélectionner la source d'horloge employée par le Timer/Compteur0 :

CS02	CS01	CS00	Source
0	0	0	0 (stop le compteur)
0	0	1	Horloge système / 1
0	1	0	Horloge système / 8
0	1	1	Horloge système / 64
1	0	0	Horloge système / 256
1	0	1	Horloge système / 1024
1	1	0	Broche T0, active sur front descendant
1	1	1	Broche T0, active sur front montant

Si le mode externe est employé pour le Timer/Compteur0, la transmission de l'horloge externe sur la broche **T0** se fera même si la broche est configurée en sortie. Cette particularité permet le contrôle du compteur par logiciel.

### Registre **TCNT0** (Timer/Counter Register)

Le registre **TCNT0** du Timer/Compteur0 est accessible en lecture / écriture en permanence.

Adresse	7	6	5	4	3	2	1	0
\$32	<b>TCNT7</b>	<b>TCNT6</b>	<b>TCNT5</b>	<b>TCNT4</b>	<b>TCNT3</b>	<b>TCNT2</b>	<b>TCNT1</b>	<b>TCNT0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**TCNT7 :0 Timer/Counter Register** Valeur du Timer/Compteur0 lisible en permanence et modifiable à volonté, mais si le mode comparaison est actif, le contrôle peut être passé et entraîner une erreur dans votre programme, le compteur ne se bloquera pas.

## Registre OCR0 (Output Compare Register)

Le registre **OCR0** du Timer/Compteur0 est accessible en lecture / écriture en permanence.

Adresse	7	6	5	4	3	2	1	0
\$32	<b>OCR07</b>	<b>OCR06</b>	<b>OCR05</b>	<b>OCR04</b>	<b>OCR03</b>	<b>OCR02</b>	<b>OCR01</b>	<b>OCR00</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCR07 :0 Output Compare Register** La valeur du registre est continuellement comparée avec la valeur de **TCNT0**. Si les deux valeurs sont identiques, cela peut produire une interruption ou une forme d'onde sur la broche **OC0** selon la configuration choisie.

## Registre TIFR (Timer/Counter Interrupt Flag)

Le registre **TIFR** indique l'état des interruptions internes du Timer0 et du Comparateur. Les interruptions seront prises en charge si le bit correspondant dans le registre **TIMSK** à été activé (mis à 1).

Adresse	7	6	5	4	3	2	1	0
\$38	OCT2	TOV2	ICF1	OCF1A	OCF1B	TOV1	<b>OCF0</b>	<b>TOV0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCF0 Output Compare Flag 0** Le bit est mis à 1 quand une arrivée est comparée entre le Timer/Counter0 et les données du registre **OCR0**. **OCF0** est remis à 0 par le matériel lors de l'exécution de l'interruption. Vous pouvez aussi mettre le bit à 0.

**TOV0 Timer/Counter0 Overflow Flag** Le bit est mis à 1 quand le Timer/Counter0 déborde. **TOV0** est automatiquement remis à 0 quand l'interruption est exécutée. Vous pouvez aussi mettre le bit à 0. Dans le mode **PWM**, ce bit est mis à 1 quand le Timer/Counter0 arrive à la valeur **\$00**.

## Registre TIMSK (Timer/Counter Interrupt Mask)

Le registre **TIMSK** définit individuellement le masque pour les modules Timer, Compteurs et Comparateurs. Le masque autorise les interruptions s'il est à 1, s'il est à 0 l'interruption ne sera pas prise en compte. Lors de l'interruption, le registre **TIFR** contiendra le bit du module qui à activer l'interruption et qu'il faudra tester pour connaître le modules en cause.

Adresse	7	6	5	4	3	2	1	0
\$39	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	<b>OCIE0</b>	<b>TOIE0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCIE0 Timer/Counter0 Output Compare Match Interrupt Enable** Quand le bit est à 1 l'interruption est validé pour le Timer/Counter0 et le bit **OCF0** est modifié dans le registre **TIFR**.

**TOIE0 Timer/Counter0 Overflow Interrupt Enable** Quand le bit est à 1 l'interruption est validé pour le Timer/Counter0 et le bit **TOV0** est modifié dans le registre **TIFR**.

## Programmation du Timer/Compteur0

### Paramétrage du TIMER 0

Si interruption penser à mettre l'instruction **SEI**

Dans le même cas, mettre à 1 le drapeau **TOIE0** du registre **TIMSK**

Inscription de la valeur du compteur avec **TCNT0**

Sélection de la source d'horloge du compteur dans **TCCR0**, qui met le compteur en fonctionnement

Exécution de la routine d'interruption et rechargement du compteur.

### Exemple de programmation

En premier lieu, la routine d'interruption du Timer0.

```
TIM0_OVF:   in      sauvreg,sreg      ; sauvegarde de sreg
            Push   sauvreg          ; sreg est poussé dans la pile
            push  a                  ; a est poussé sur la pile
            ldi   a,$9C              ; rechargement de la valeur 156
            out   tcnt0,a            ; dans le registre du compteur
            sbi   portb,0            ; PB0 =1
```

```

nop
nop
nop
nop
nop
nop
nop ; Nop pour allonger l'impulsion
cbi portb,0 ; PB0 =0
pop a ; a est restitué
pop sauvreg ; sauvreg est restitué
out sreg,sauvreg ; sreg est remis dans son contexte (avant interruption)
reti ; fin de l'interruption
;Programme de d'Initialisation
RESET: ldi a,low(RAMEND)
out SPL,a ; Initialisation de la pile à
ldi a,high(RAMEND) ; l'adresse haute de la SRAM
out SPH,a
ldi a,$FF ; port b en sortie
out ddrb,a
ldi r16,$00
out portb,a ; sortie = 0
sei ; autorisation interruption général
;Programme de Début
debut: ldi a,$01
out tmsk,a ; Validation de l'interruption.
ldi a,$9C ; chargement de la valeur 156
out tcnt0,a ; dans le registre du compteur
ldi a,$04
out tccr0,a ; Sélection de la fréquence de pré division
debu1: ...

```



## Mode de fonctionnement du Timer1

**Remarque :** pour l'accès aux registres à 16 bits **OCR1A/B** et **ICR1** via le bus de données à 8 bits, en mode Ecriture l'octet de poids fort doit être réalisé en premier, suivie de l'écriture du poids faible. En mode Lecture c'est l'inverse, l'octet de poids faible doit être lu en premier suivi de la lecture de l'octet de poids fort comme le montre le petit programme d'exemple qui suit :

```
; Mettre TCNT1 à $01FF
Ldi  r17,$01
Ldi  r16,$FF
Out  TCNT1H,r17      ; Toujours en premier en Ecriture
Out  TCNT1L,r16
; Lire TCNT1 et transfert des valeurs dans r17:r16
in   r16,TCNT1L     ; Toujours en premier en Lecture
in   r17,TCNT1H
```

Le Timer/Compteur1 peut être cadencé **clkT1** par le pré diviseur ou une source d'horloge externe sur la broche **T1 (PB1)**. Le compteur est incrémenter (ou décrémente) sur une plage de 65 536 valeurs (16 bits). Le Timer/Compteur1 est inactif quand aucune source d'horloge n'est choisie.

La source de capture d'entrée **ICP** peut employer le comparateur analogique comme déclencheur. L'échantillonnage employé est le même que l'entrée **T1**. Le détecteur de front est identique, par contre un supprimeur de bruit complémentaire est inséré avant le détecteur de front, qui augmente le retard de quatre cycles d'horloge système. Notez que supprimeur de bruit et le détecteur de front sont toujours en service sauf dans le cas le Timer1 est en mode générateur de forme d'onde avec **ICR1** pour **Max**. La capture d'entrée peut être déclenchée par logiciel avec l'entrée **ICP** et par le comparateur analogique (Voir le chapitre sur le comparateur analogique).

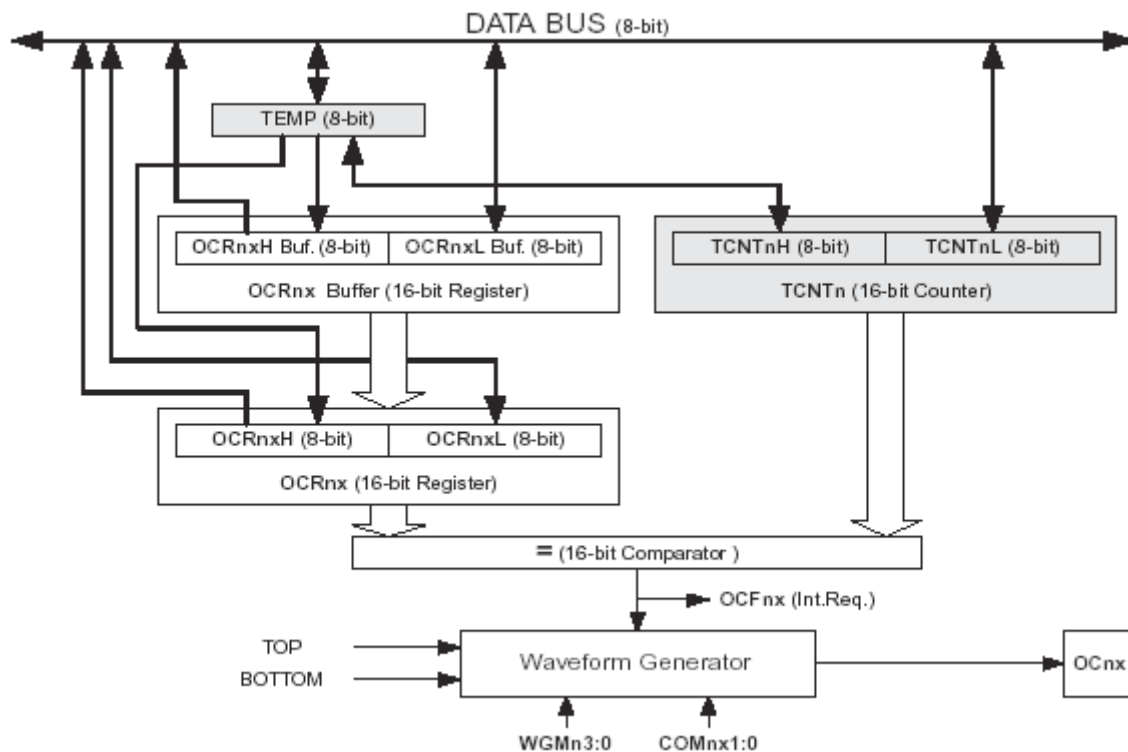


Figure 24, Synoptique du comparateur à 16 bits.

Le registre **TCNT1** est comparé en permanence au registre **OCR1A/B** et le résultat peut générer des formes d'onde **PWM** ou des fréquences variables sur les broches **OC1A/B**. La comparaison mettra à jour le drapeau **OCF1A/B** qui peut être employé pour produire une demande d'interruption.

La valeur maximale du Timer/Compteur1, définie par le mode de fonctionnement, peut être le registre **OCR1A**, le registre **ICR1**, ou une valeur fixe définie par le choix du mode **PWM rapide**.

**Restriction :** le registre **OCR1A** ne peut pas être employé pour produire une forme d'onde **PWM**, le registre **ICR1** peut être une alternative.

Le drapeau **OCF1A/B** est positionné sur l'égalité de comparaison sur le cycle d'horloge suivant. L'interruption est demandée si **OCIE1A/B** est à 1. Le drapeau **OCF1A/B** est automatiquement mis à 0 quand l'interruption est exécutée. Autrement, le drapeau **OCF1A/B** peut être mis à 0 par le logiciel.

Le générateur de forme d'onde emploie le signal de comparaison pour produire selon le mode des bits **WGM13:10** et le mode de front des bits **COM1A/B1:0**. C'est le Générateur de Forme d'Onde qui traite le cas des valeurs extrêmes du compteur (\$00 et Max). Dans la figure qui suit, le registre de contrôle du port d'entrée-sortie **DDR** et **PORT** est affecté par les bits **COM1A/B1:0**. Lors d'un **RESET** la sortie **OC1A/B** est mise à 0.

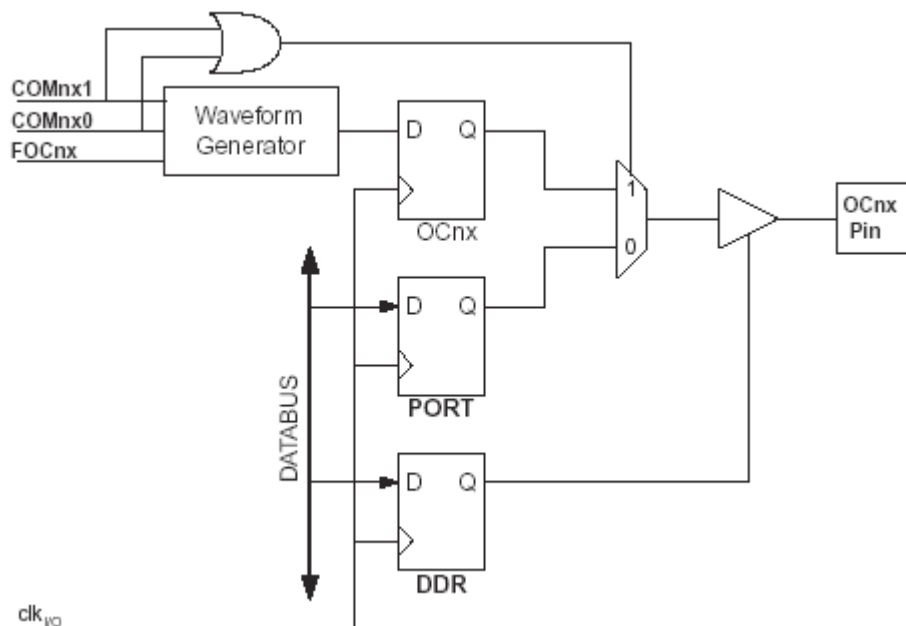


Figure 25, Synoptique de la mise en forme d'onde sur **OC1A/B**.

Comme vous avez pu le voir, le Timer1 est particulièrement complet et peut être utilisé pour une multitude d'application, nous allons entrer en détaille dans les 5 modes de fonctionnement de base.

### Le mode Normal du Timer/Compteur1

Le mode normal est le mode de fonctionnement le plus simple (**WGM13:10** à 0). Dans ce mode le comptage est toujours incrémental sans remise à 0 du compteur. Il passe par le maximal **\$FFFF** en débordant, le drapeau **TOV1** est mis à 1, puis le compteur repart au début **\$0000**. Le drapeau **TOV1** se comporte comme un 17ème bit mais il n'est pas remis à 0. Cependant, combiné avec l'interruption de débordement du Timer, le drapeau **TOV1** peut être remis à 0 automatiquement, augmentant d'un bit la résolution du Timer. Une nouvelle valeur peut être écrite n'importe quand dans le registre **TCNT1**.

Le Timer combiné avec l'unité de comparaison **OCR1A/B** peut être employée pour produire une interruption au bout d'un temps donnée. L'utilisation de cette possibilité n'est pas recommandée pour la production de forme d'onde, puisque cela occupera trop de temps **MPU**, le mode suivant est préférable pour cela.

### Le mode Timer1 à Remise à 0 sur Comparaison (CTC)

Le mode **CTC** (**WGM13:10** à 4 ou 12), le registre **OCR1A** ou **IRC1** est employé pour définir la résolution du compteur. Le compteur est incrémentée et quant la valeur est égale à **OCR1A** (**WGWM13:10** = 4) ou **ICR1** (**WGM13:10** = 12), le compteur est remis à 0 et le drapeau **OCF1A** ou **ICF1** est mis à 1 pour déclencher une éventuelle interruption. **L'OCR1A** ou **ICR1** définit la valeur supérieure pour le compteur donc sa résolution.

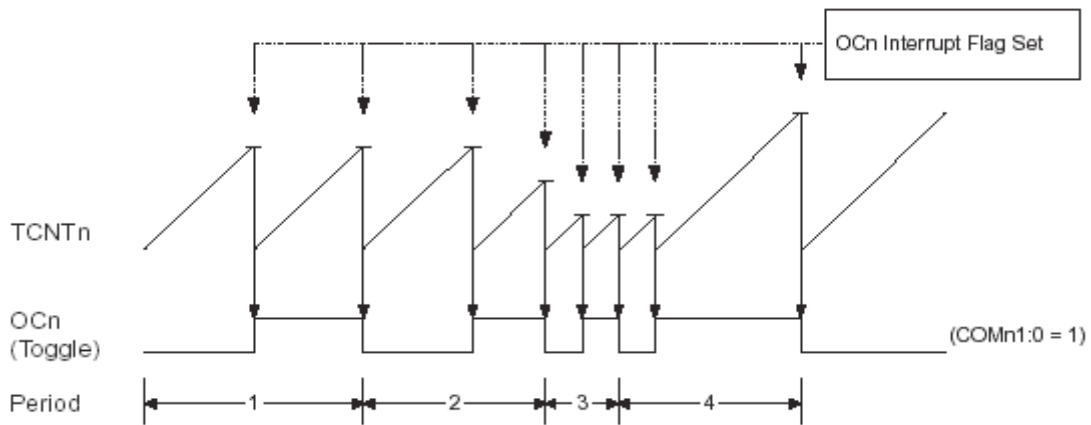


Figure 26, Mode **CTC** à remise à 0.

Le drapeau **OCF1A** ou **ICR1** est automatiquement remis à 0 lors de l'exécution de l'interruption. Ce mode permet de mieux contrôler la fréquence produite. Il simplifie aussi l'opération de comptage des événements externes. La fréquence maximum que peut produire le **CTC** est fonction de la formule suivante :

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

Avec **N** le rapport du pré diviseur (1, 8, 64, 256 ou 1024), **Fclk\_I/O** étant la fréquence d'horloge du quartz.

### Le mode Modulation en Largeur d'Impulsion Rapide (PWM)

Le mode **PWM rapide** permet de produire des impulsions de largeur variable en fonction des informations du registre **WGM13:10** à 5, 6, 7, 14 & 15. Le compteur est incrémenté de **\$0000** à **\$FFFF** puis reprend à **\$0000**. Le bit **OC1 (PB3)** est mis à 1 au début du comptage puis il est mis à 0 lors de la comparaison entre **TCNT1** et **OCR1A/B** ou **ICR1**. La remise à zéro du compteur **TCNT0** doit être faite dans la gestion d'interruption pour une période plus courte. Il est possible d'inverser le niveau **OC1A/B** par les bits **COM1:0**. La forme d'onde n'est pas symétrique.

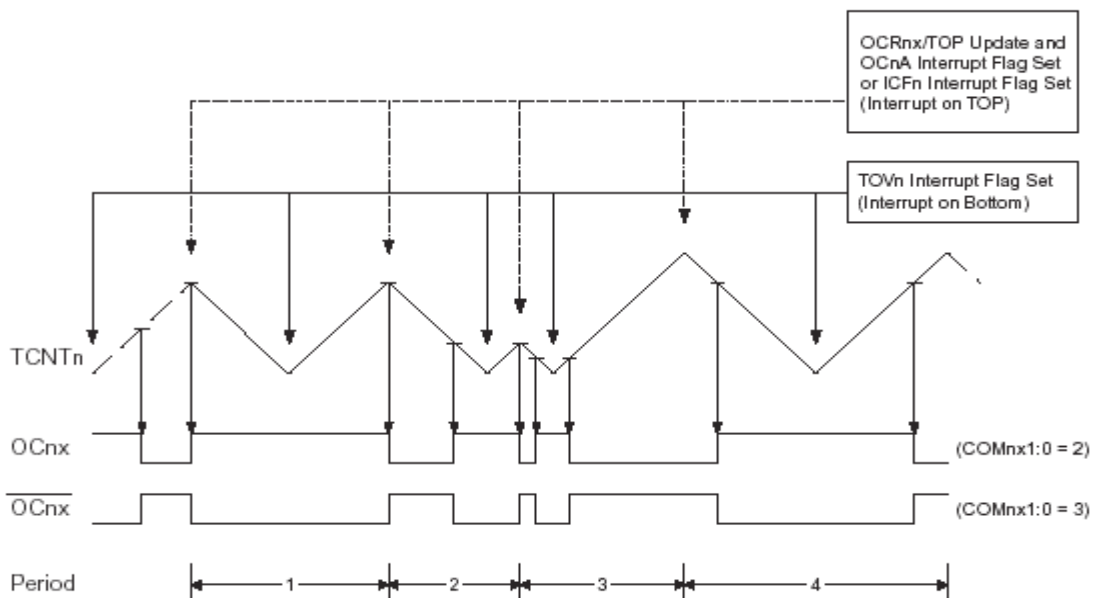


Figure 27, Mode **PWM Rapide** sur simple pente montante.

La résolution peut être fixée à 8, 9 ou à 10 bits, ou bien être définie par **ICR1** ou **OCR1A/B**. La résolution minima permise est de 2 bits (**ICR1** ou **OCR1A/B** = **\$0003**) et la résolution maximale est de 16 bits (**ICR1** ou **OCR1A/B** = **\$FFFF**). La résolution **PWM** peut être calculée en employant l'équation suivante :

$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)}$$



La valeur **TOP** dans le mode **PWM rapide** correspond à la valeur : fixes \$00FF, \$01FF ou \$03FF avec respectivement **WGM13:10** à **5, 6, ou 7**, dans **ICR1** avec **WGM13:10** à **14**, ou dans **OCR1A** avec **WGM13:0** à **15**. La fréquence maximum que peut produire le mode **PWM rapide** est fonction de la formule suivante :

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

Avec **N** le rapport du pré diviseur (1, 8, 64, 256 ou 1024) avec **OCR1A** ou **ICR1** = \$0003 pour le maximum de vitesse, **Fclk\_I/O** étant la fréquence d'horloge du quartz.

### Le mode PWM Correct

Le mode **PWM correct** utilise le compteur dans les deux sens, quant la valeur définie dans le registre **OCR1A** est atteinte la sortie **OC1A/B** est changé et le compteur repart à l'envers. Quant il arrive à \$0000, la valeur d'**OC1A/B** est à nouveau changée. Le mode **PWM correct** est défini par **WGM13:10** à **1, 2, 3 10 ou 11**, fournit une phase de haute résolution de forme d'onde **PWM**. Le **PWM correct** est basée sur une opération à double pente. Le compteur change de direction, il passe de \$0000 à Max et ensuite de Max à \$0000. Dans le mode montant la sortie **OC1A/B** est remise à 0 sur comparaison entre **TCNT1** et **OCR1A** ou **ICR1** et dans le mode descendant, la sortie sur **OC1A/B** est mis à 1.

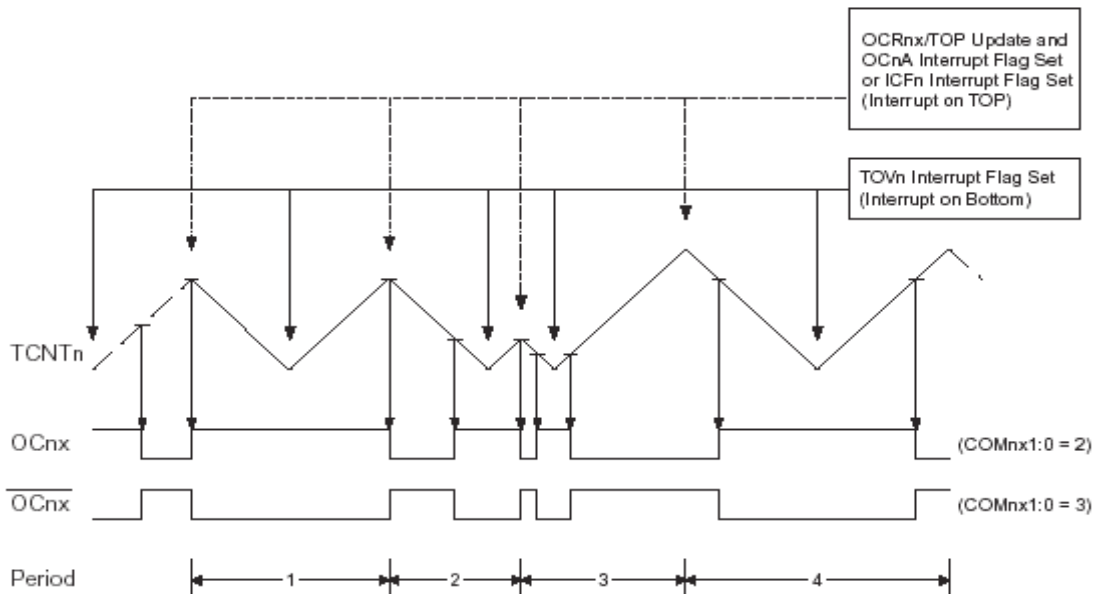


Figure 28, Mode PWM Correcte sur double pente.

La résolution **PWM correct** peut être fixée à 8, 9 ou à 10 bits, ou bien définie par **ICR1** ou **OCR1A**. La résolution minima permise est de 2 bits (**ICR1** ou **OCR1A** = \$0003) et la résolution maximale est de 16 bits (**ICR1** ou **OCR1A** au Max). La résolution **PWM** peut être calculée en employant l'équation suivante :

$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

La valeur **TOP** dans le mode **PWM correct** correspond à la valeur : fixes \$00FF, \$01FF ou \$03FF avec respectivement **WGM13:10** à **1, 2, ou 3**, dans **ICR1** avec **WGM13:10** à **10**, ou dans **OCR1A** avec **WGM13:10** à **11**. La fréquence est plus basse en **PWM correct** mais présente l'avantage d'avoir un signal plus propre, préféré pour des applications de contrôle du moteur. La valeur de **TCNT1** sera égale à deux fois **TOP** pour un cycle d'horloge de Timer. La fréquence maximum que peut produire le **PWM correct** est fonction de la formule suivante :

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

Avec **N** le rapport du pré diviseur (1, 8, 64, 256 ou 1024), **Fclk\_I/O** étant la fréquence d'horloge du quartz.

## Le mode PWM à Fréquence Corrigée

Le mode **PWM à Fréquence corrigée** **WGM13:10** à 8 ou 9 fournit une phase de haute résolution avec une fréquence corrigée. Elle est basée sur une opération à double rampe. Le compteur part de \$0000 puis est incrémenté à chaque top d'horloge, arrivé à la comparaison positive au sommet, le comptage est inversé de haut en bas, arrivée à \$0000, il repart pour une nouvelle session. Dans l'ascension **OC1A/B** est mis à 0 et quand la comparaison entre **TCNT1** et **OCR1A** est égale, **OC1A** est mis à 1. La différence principale entre le mode **PWM correct** et le mode **PWM à Fréquence corrigée** est que le registre **OCR1A** est mis à jour afin de corriger le décalage des commutations.

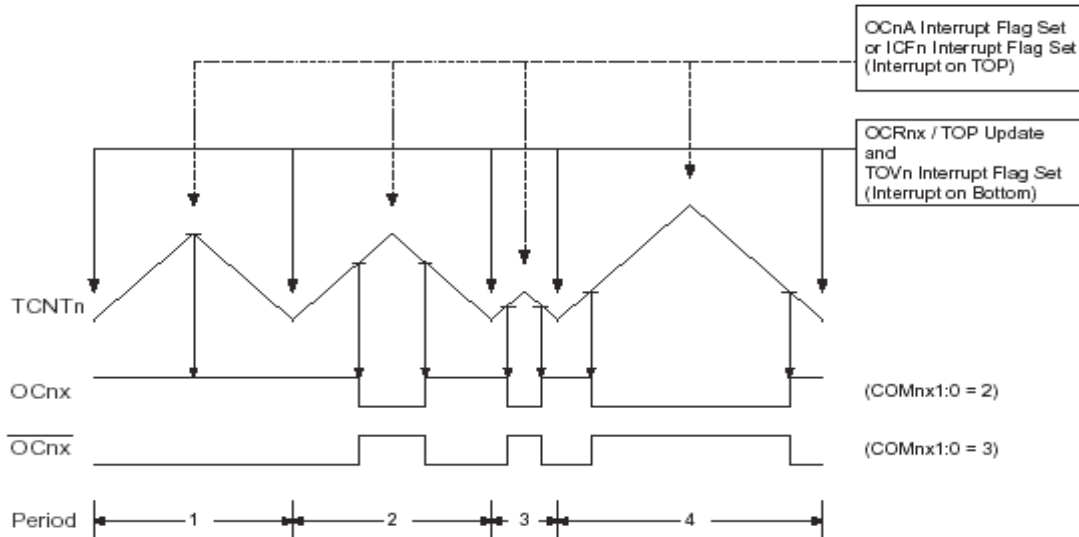


Figure 29, PWM à Fréquence corrigée.

La résolution **PWM à Fréquence corrigée** peut être définie par **ICR1** ou **OCR1A** avec une résolution minimale permise de **ICR1** ou **OCR1A** = \$0003 et une résolution maximale sur 16 bits **ICR1** ou **OCR1A** à \$FFFF-1. La résolution **PWM** peut être calculée avec l'équation suivante :

$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

La valeur **TOP** dans le mode **PWM à Fréquence corrigée** correspond à la valeur : dans **ICR1** avec **WGM13:10** à 8, ou dans **OCR1A** avec **WGM13:10** à 9.

Le drapeau d'interruption **OC1A/B** est placé sur une comparaison active. Le drapeau de débordement **TOV1** est mis à 1 sur le même cycle d'horloge que la mise à jour d'**OCR1A/B** avec la valeur double du **TOP**. Quand **OCR1A** ou **ICR1** est employée pour définir la valeur maximum, le drapeau **ICF1** est activé quand **TCNT1** a atteint le sommet. Les drapeaux d'interruption peuvent alors être employés pour produire une demande d'interrompu à chaque fois que le compteur atteint la valeur \$0000 ou le sommet (Max). En chargeant la valeur Max de sommet le programme doit vérifier que la nouvelle valeur est supérieure ou égale à la valeur de tous les comparateurs sinon le compteur n'arrivera jamais entre le **TCNT1** et l'**OCR1A/B** et le timer1 ne fonctionnera pas.

La mode **PWM à Fréquence corrigée** produit une onde symétrique. Puisque les **OCR1A** est mis à jour, la longueur de la pente montante et la pente descendante sont toujours égales. Cela donne des impulsions de production symétriques et est donc une fréquence correcte.

Le forme d'onde produite est fonction des bits **COM1A/B1:0**, mis à 2 cela produit une onde **PWM** normale et mis à 3 cela produit une onde **PWM** inversée. La fréquence **PWM à Fréquence corrigée** peut être calculé par l'équation suivante :

$$f_{OCnxPFCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

Avec **N** le rapport du pré diviseur (1, 8, 64, 256 ou 1024), **Fclk\_I/O** étant la fréquence d'horloge du quartz.

Les valeurs extrêmes d'**OCR1A/B** représentent des cas spéciaux qu'il faut éviter, le minimum est normalement de \$0003 et le maximum de \$FFFF-1 (soit \$FFFE).

## Les Registres Associés au Timer/Compteur1

Le Timer1 utilise 5 registres spécifiques et deux registres d'interruption.

### Registre TCCR1A (Timer/Counter1 Control Register A)

Le registre de control du Timer/Compteur1 partie A, le choix des modes utilise aussi le registre **TCCR1B**.

Adresse	7	6	5	4	3	2	1	0
\$2F	<b>COM1A1</b>	<b>COM1A0</b>	<b>COM1B1</b>	<b>COM1B0</b>	<b>FOC1A</b>	<b>FOC1B</b>	<b>WGM11</b>	<b>WGM10</b>
L/E	L/E	L/E	L/E	L/E	E	E	L/E	L/E

**COM1A1&0** Compare Output Mode For Channel A ; voir ci-dessous **COM1B1&0**.

**COM1B1&0** Compare Output Mode For Channel B Ces bits contrôlent la production de comparaison sur les sorties **OC1A** et **OC1B** respectivement. Si un ou les deux bits sont écrits à 1, la sortie **OC1A/B** ignore la fonctionnalité normale d'entrée-sortie à laquelle il est connecté. Cependant, notez que le registre de direction des données **DDR** doit être placé en sortie. Le tableau qui suit précise le fonctionnement des sorties **OC1A/B** en fonction du mode choisi dans **WGM13:10** qui doit être **Normal** ou **CTC**, mais pas de mode PWM.

<b>COM1A1</b> <b>COM1B1</b>	<b>COM1A0</b> <b>COM1B0</b>	Description
0	0	Opération de port Normale, <b>OC1A/OC1B</b> débranché
0	1	Interrupteur à bascule <b>OC1A/OC1B</b> sur comparaison
1	0	Mis à 0 d' <b>OC1A/OC1B</b> sur comparaison
1	1	Mis à 1 d' <b>OC1A/OC1B</b> sur comparaison

Dans le mode **PWM rapide uniquement** :

<b>COM1A1</b> <b>COM1B1</b>	<b>COM1A0</b> <b>COM1B0</b>	Description
0	0	Opération de port Normal, <b>OC1A/OC1B</b> débranché.
0	1	<b>WGM13:10 à 15</b> Interrupteur à bascule <b>OC1A</b> sur Comparaison, <b>OC1B</b> débranché (Opération de port normal). Pour les autres modes de <b>WGM13:10</b> , opération de port normal, <b>OC1A/OC1B</b> débranché.
1	0	Sur comparaison au sommet, mis à 1 d' <b>OC1A/OC1B</b>
1	1	Sur comparaison au sommet, mise à 0 d' <b>OC1A/OC1B</b>

Dans le mode **PWM correcte et à Fréquence uniquement** :

<b>COM1A1</b> <b>COM1B1</b>	<b>COM1A0</b> <b>COM1B0</b>	Description
0	0	Opération de port Normale, <b>OC1A/OC1B</b> débranché.
0	1	<b>WGM13:10 à 9 ou 14</b> Interrupteur à bascule <b>OC1A</b> sur Comparaison, <b>OC1B</b> débranché (Opération de port normal). Pour les autres mode <b>WGM13:10</b> , opération de port normal, <b>OC1A/OC1B</b> débranché.
1	0	<b>OC1A/OC1B</b> Mis à 0 sur comparaison en montant. <b>OC1A/OC1B</b> Mis à 1 sur comparaison en descendant.
1	1	<b>OC1A/OC1B</b> Mis à 1 sur comparaison en montant. <b>OC1A/OC1B</b> Mis à 0 sur comparaison en descendant.

**FOC1A/B** Compare Output Mode For Channel B Ces bits sont actifs seulement quand un mode non **PWM** est sélectionné. Cependant, pour assurer la compatibilité avec des dispositifs futurs, ces bits doivent être mis à 0 quand **TCCR1A** est écrit dans un mode **PWM**. **FOC1A** commande **OC1A** et **FOC1B** commande **OC1B**. En écrivant un 1 dans **FOC1A/B**, l'état de la comparaison est imposée à l'unité de Génération de Forme d'Onde et l'état des sorties **OC1A/B** est changée selon les bits **COM1A/B1:0**. C'est une fonction de type **strobe**. Un **FOC1A/B strobe** ne produira pas

d'interruption, donc il ne mettra pas à 0 le compteur dans le mode **CTC** avec **OCR1A** comme sommet. Les bits **FOC1A/B** sont toujours lues à 0.

**WGM11&10 Waveform Generation Mode** Combiné avec les bits **WGM13&12** situées dans le registre **TCCR1B**, ces bits contrôlent : l'ordre de compte, la source de la valeur maximale et le genre de génération de forme d'onde à employer. Les 15 modes de fonctionnement du Timer1 sont repris dans le tableau qui suit (le mode 13 n'est pas utilisable) :

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Mode du Timer1	OCR1A/B Modifier sur	Drapeau TOV1 sur
0	0	0	0	0	Normal	Immédiate	\$FFFF
1	0	0	0	1	PWM correct à 8-bit	\$00FF	\$00
2	0	0	1	0	PWM correct à 9-bit	\$01FF	\$00
3	0	0	1	1	PWM correct à 10-bit	\$03FF	\$00
4	0	1	0	0	CTC	Immédiate	OCR1A
5	0	1	0	1	PWM rapide à 8-bit	\$00FF	\$00FF
6	0	1	1	0	PWM rapide à 9-bit	\$01FF	\$01FF
7	0	1	1	1	PWM rapide à 10-bit	\$03FF	\$03FF
8	1	0	0	0	PWM à Fréquence Corrigée avec <b>ICR1</b>	\$00	\$00
9	1	0	0	1	PWM à Fréquence Corrigée avec <b>OCR1A</b>	\$00	\$00
10	1	0	1	0	PWM Correct	ICR1	\$00
11	1	0	1	1	PWM Correct	OCR1A	\$00
12	1	1	0	0	CTC	Immediate	ICR1
13	1	1	0	1	Réservé	-	-
14	1	1	1	0	PWM rapide	ICR1	ICR1
15	1	1	1	1	PWM rapide	OCR1A	OCR1A

### Registre TCCR1B (Timer/Counter1 Control Register B)

Le registre de control du Timer/Compteur1 partie B, le choix des modes utilise aussi le registre **TCCR1A**.

Adresse	7	6	5	4	3	2	1	0
\$2E	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
L/E	L/E	L/E	L	L/E	L/E	L/E	L/E	L/E

**ICNC1 Input Capture Noise Canceler** Ce bit mis à 1 active le supprimeur de bruit sur l'entrée de capture. Quand le supprimeur de bruit est activé, l'entrée **ICP1** est filtrée. La fonction de filtre exige quatre échantillons successifs estimés égaux pour que la valeur soit pris en charge, ce qui engendre donc un retard de quatre cycles d'horloge du Timer1.

**ICES1 Input Capture Edge Select** Ce bit détermine le front sur lequel l'entrée **ICP1** est employé pour déclencher un événement de capture. Quand le bit est mis à 0, un front descendant (négatif) est employé comme déclencheur et quand il est mis à 1, un front montant (positif) déclenchera la capture. Quand une capture est déclenchée selon **ICES1**, la valeur inverse (complémentaire) est copiée dans le registre de capture **ICR1**. L'événement mettra aussi le drapeau **ICF1** à 1 et cela peut être employé pour faire une demande d'interruption. Quand **ICR1** est employé comme valeur supérieure, **ICP1** est débranché et par conséquent la fonction de capture d'entrée est mise hors de service.

**WGM13:12 Waveform Generation Mode** Voir le registre **TCCR1A**.

**CS12:10 Clock Select** Les trois bits de choix de l'horloge permettent de sélectionner la source d'horloge employée par le Timer/Compteur1 :

CS02	CS01	CS00	Source
0	0	0	0 (stop le compteur)
0	0	1	Horloge système / 1
0	1	0	Horloge système / 8
0	1	1	Horloge système / 64
1	0	0	Horloge système / 256
1	0	1	Horloge système / 1024
1	1	0	Broche T1, active sur front descendant
1	1	1	Broche T1, active sur front montant

Si le mode externe est employé pour le Timer/Compteur1, la transmission de l'horloge externe sur la broche **T1** se fera même si la broche est configurée en sortie. Cette particularité permet le contrôle du compteur par logiciel.

### Registre TCNT1H & TCNT1L (Timer/Counter1 Register)

Les deux registres **TCNT1H** et **TCNT1L**, combiné dans **TCNT1** donnent l'accès direct au compteur à 16 bits du Timer1. Les octets hauts et bas sont lus et écrits simultanément par l'intermédiaire d'un registre provisoire à 8 bits pour les octets hauts. Ce registre provisoire est partagé par tous les registres à 16 bits. La modification du compteur **TCNT1** tandis que le compteur court présente un risque de manquer une comparaison entre **TCNT1** et **OCR1A/B**.

Adresse	15	14	13	12	11	10	9	8
\$2D	<b>TCNT1H7</b>	<b>TCNT1H6</b>	<b>TCNT1H5</b>	<b>TCNT1H4</b>	<b>TCNT1H3</b>	<b>TCNT1H2</b>	<b>TCNT1H1</b>	<b>TCNT1H0</b>
Adresse	7	6	5	4	3	2	1	0
\$2C	<b>TCNT1L7</b>	<b>TCNT1L6</b>	<b>TCNT1L5</b>	<b>TCNT1L4</b>	<b>TCNT1L3</b>	<b>TCNT1L2</b>	<b>TCNT1L1</b>	<b>TCNT1L0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**TCNT1H&L Timer/Counter1 Register** C'est le contenu du compteur du Timer1 sur 16 bits.

### Registre OCR1AH and OCR1AL (Output Compare Register 1 A)

Les deux registres **OCR1AH** et **OCR1AL**, combiné dans **OCR1A** donnent l'accès direct au comparateur A à 16 bits du Timer1.

Adresse	15	14	13	12	11	10	9	8
\$2B	<b>OCR1AH7</b>	<b>OCR1AH6</b>	<b>OCR1AH5</b>	<b>OCR1AH4</b>	<b>OCR1AH3</b>	<b>OCR1AH2</b>	<b>OCR1AH1</b>	<b>OCR1AH0</b>
Adresse	7	6	5	4	3	2	1	0
\$2A	<b>OCR1AL7</b>	<b>OCR1AL6</b>	<b>OCR1AL5</b>	<b>OCR1AL4</b>	<b>OCR1AL3</b>	<b>OCR1AL2</b>	<b>OCR1AL1</b>	<b>OCR1AL0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCR1AH&L Output Compare Register 1 A** C'est le contenu du comparateur A du Timer1 sur 16 bits qui est continuellement comparée avec la valeur **TCNT1** et produit de forme d'onde sur la broche **OC1A** lors de l'égalité.

### Registre OCR1BH and OCR1BL (Output Compare Register 1 B)

Les deux registres **OCR1BH** et **OCR1BL**, combiné dans **OCR1B** donnent l'accès direct au comparateur B à 16 bits du Timer1.

Adresse	15	14	13	12	11	10	9	8
\$29	<b>OCR1BH7</b>	<b>OCR1BH6</b>	<b>OCR1BH5</b>	<b>OCR1BH4</b>	<b>OCR1BH3</b>	<b>OCR1BH2</b>	<b>OCR1BH1</b>	<b>OCR1BH0</b>
Adresse	7	6	5	4	3	2	1	0
\$28	<b>OCR1BL7</b>	<b>OCR1BL6</b>	<b>OCR1BL5</b>	<b>OCR1BL4</b>	<b>OCR1BL3</b>	<b>OCR1BL2</b>	<b>OCR1BL1</b>	<b>OCR1BL0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCR1BH&L Output Compare Register 1 B** C'est le contenu du comparateur B du Timer1 sur 16 bits qui est continuellement comparée avec la valeur **TCNT1** et produit de forme d'onde sur la broche **OC1B** lors de l'égalité.

### Registre ICR1H and ICR1L (*Input Compare Register 1*)

Les deux registres **ICR1H** et **ICR1L**, combiné dans **ICR1** donnent l'accès direct à l'entrée de capture **ICP1** à 16 bits du Timer1.

Adresse	15	14	13	12	11	10	9	8
\$27	<b>ICR1H7</b>	<b>ICR1H6</b>	<b>ICR1H5</b>	<b>ICR1H4</b>	<b>ICR1H3</b>	<b>ICR1H2</b>	<b>ICR1H1</b>	<b>ICR1H0</b>
Adresse	7	6	5	4	3	2	1	0
\$26	<b>ICR1L7</b>	<b>ICR1L6</b>	<b>ICR1L5</b>	<b>ICR1L4</b>	<b>ICR1L3</b>	<b>ICR1L2</b>	<b>ICR1L1</b>	<b>ICR1L0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**ICR1H&L** *Input Compare Register 1* C'est le contenu de l'entrée de capture du Timer1 sur 16 bits qui est continuellement comparée avec la valeur **TCNT1** et produit de forme d'onde sur la broche **OC1A/B** lors de l'égalité.

### Registre TIFR (*Timer/Counter Interrupt Flag*)

Le registre **TIFR** indique l'état des interruptions internes des modules Timer et Comparateur. Les interruptions seront prises en charge si le bit correspondant dans le registre **TIMSK** à été activé (mis à 1).

Adresse	7	6	5	4	3	2	1	0
\$38	<b>OCT2</b>	<b>TOV2</b>	<b>ICF1</b>	<b>OCF1A</b>	<b>OCF1B</b>	<b>TOV1</b>	<b>OCF0</b>	<b>TOV0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**ICF1** *Timer/Counter1, Input Capture Flag* Ce bit est mis à 1 quand un événement de capture arrive sur la broche **ICP1**. Quand le registre de capture **ICR1** est utilisé, le bit est mis à 1 par l'arrivée d'une valeur supérieure à **ICR1**. **ICF1** est automatiquement remis à 0 quand l'interruption est exécutée. Vous pouvez aussi forcer **ICF1** à 0.

**OCF1A** *Timer/Counter1, Output Compare A Match Flag* Ce bit est modifié après que le cycle d'horloge du minuteur est atteint la valeur correspondant au registre **TCNT1**. Notez qu'une comparaison forcée en sortie avec **FOC1A** ne mettra pas le bit à un. **OCF1A** est automatiquement remis à 0 lors de l'exécution de l'interruption. Vous pouvez aussi forcer **OCF1A** à 0.

**OCF1B** *Timer/Counter1, Output Compare B Match Flag* Ce bit est modifié après que le cycle d'horloge du minuteur est atteint la valeur du registre **TCNT1**. Notez qu'une comparaison forcée en sortie avec **FOC1B** ne mettra pas le bit à un. **OCF1B** est automatiquement remis à 0 quand l'interruption est exécutée. Vous pouvez aussi mettre le bit à 0.

**TOV1** *Timer/Counter1, Overflow Flag* La mise à 1 de ce bit dépend du mode de fonctionnement défini pour le Timer/Compteur1. Dans le mode normal et **CTC**, le bit **TOV1** est mis à 1 quand le minuteur déborde. **TOV1** est automatiquement remis à 0 quand l'interruption est exécutée. Vous pouvez aussi mettre le bit à 0.

### Registre TIMSK (*Timer/Counter Interrupt Mask*)

Le registre **TIMSK** définit individuellement le masque pour les modules Timer, Compteurs et Comparateurs. Le masque autorise les interruptions s'il est à 1, si il est à 0 l'interruption ne sera pas prise en compte. Lors de l'interruption, le registre **TIFR** contiendra le bit du module qui à activer l'interruption et qu'il faudra tester pour connaître le modules en cause.

Adresse	7	6	5	4	3	2	1	0
\$39	<b>OCIE2</b>	<b>TOIE2</b>	<b>TICIE1</b>	<b>OCIE1A</b>	<b>OCIE1B</b>	<b>TOIE1</b>	<b>OCIE0</b>	<b>TOIE0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**TICIE1** *Timer/Counter1, Input Capture Interrupt Enable* Quand le bit est à 1 l'interruption est validé pour le Timer/Counter1 et le bit **ICF1** est modifié dans le registre **TIFR**.

**OCIE1A** *Timer/Counter1, Output Compare A Match Interrupt Enable* Quand le bit est à 1 l'interruption est validé pour le Timer/Counter1 et le bit **OCF1A** est modifié dans le registre **TIFR**.

**OCIE1B** *Timer/Counter1, Output Compare B Match Interrupt Enable* Quand le bit est à 1 l'interruption est validé pour le Timer/Counter1 et le bit **OCF1B** est modifié dans le registre **TIFR**.

**TOIE1** *Timer/Counter1, Overflow Interrupt Enable* Quand le bit est à 1 l'interruption est validé pour le Timer/Compteur1 et le bit **TOV1** est modifié dans le registre **TIFR**.

## Programmation du Timer/Compteur1

### Paramétrage du TIMER 1

Si interruption penser à mettre l'instruction **SEI**

Dans le même cas, mettre à 1 le drapeau **TOIE1** du registre **TIMSK**

Inscription de la valeur du compteur avec **TCNT1**

Sélection de la source d'horloge du compteur dans **TCCR1A/B**, qui met le compteur en fonctionnement

Exécution de la routine d'interruption et rechargement du compteur.

### Exemple de programmation

Cet exemple de capture d'entrée avec le Timer1 montrera la mise en oeuvre d'une utilisation très simple de l'événement de capture d'entrée sur interruption. Le port **PD6** est la broche de capture d'entrée **ICP**. Si le changement d'état d'**ICP** est modifié le nombre de changement sera mesuré par le Timer1. Les 8 bits les plus significatif de la valeur du Timer1 seront écrits sur le Port **B** ou sont connectés des **LEDs**, tandis que **PD6** est connecté à un commutateur. Dans cet exemple, le temps maximal détectable est d'environ une seconde **TOVCK** = 1. Avec l'équation  $F_{osc}/O / 2^{16}$ , le facteur de division d'horloge exigé, soit 64 pour ce cas et pour une horloge de système de 4 **MHz**.

Le routine d'initialisation est la suivante :

```
TIM1_CAPT:  push  r16                ; Routine de capture
            In    r16,SREG
            Push r16
            In    r16,ICR1L      ; Lire ICR bas et sauver en provisoire
            In    r16,ICR1H      ; Lire ICR haut
            Com   r16            ; complément à 1
            Out   PORTB,r16      ; Ecrire ICR1H sur PORTB
            Clr   r16
            Out   TCNT1H,r16     ; Effacer le compteur
            Out   TCNT1L,r16
            Pop   r16
            Out   SREG,r16
            Pop   r16
            Reti                 ; Fin de l'interruption
init_Ex1:   ; Initialisation Timer 1 à 16 bits
            ldi   r16,(1<<CS11)|(1<<CS10)
            out   TCCR1B,r16     ; Horloge Timer1 = horloge système / 64
            ldi   r16,1<<ICF1
            out   TIFR,r16       ; ICF1 = 0 pendant l'interruption
            ldi   r16,1<<TICIE1
            out   TIMSK,r16      ; Evénement de Capture actif
            ser   r16
            out   DDRB,r16       ; Port B en sortie
            cbi   DDRD,PD6       ; PD6/ICP en entrée
            ret
```

## Le Pré Diviseur du Timer/Compteur2

Le Timer/Compteur2 à son propre pré diviseur à 10 bits qui est présenté dans la figure suivante :

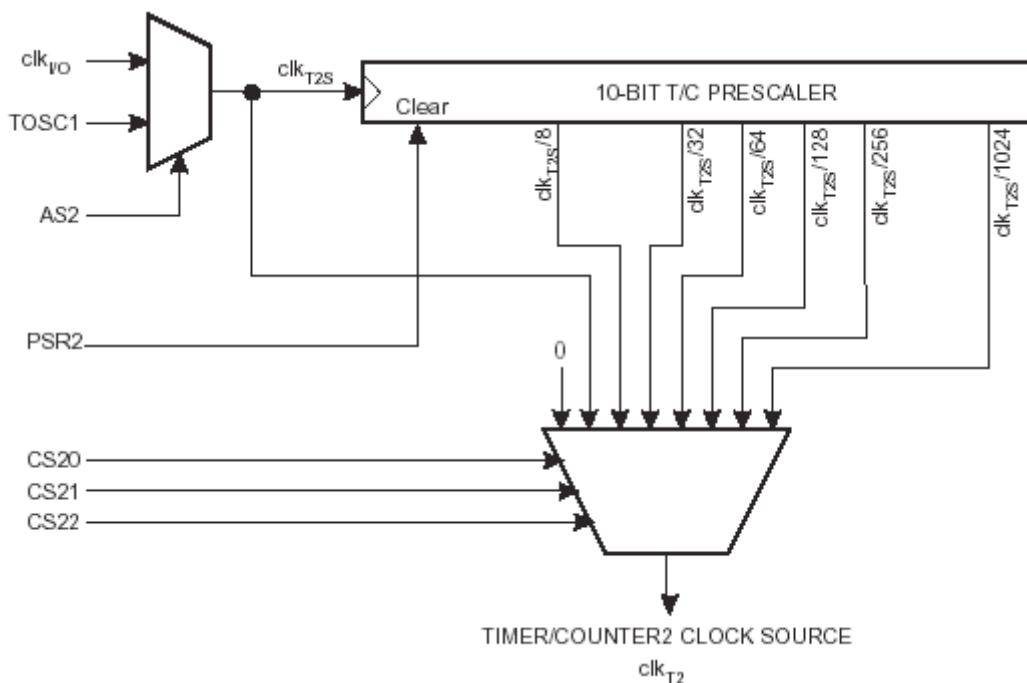


Figure 30, Synoptique du pré diviseur du Timer/Compteur2.

La source d'horloge du Timer/Compteur2 est nommée **clkT2S**. **clkT2S** est par défaut connecté à l'horloge d'entrée-sortie système **clkI/O**. En mettant le bit **AS2** à 1 dans **ASSR**, le Timer/Compteur2 est cadencé de manière asynchrone par la broche **TOSC1 (PC6)**. Cela permet l'utilisation de Timer/Compteur2 comme un compteur 'Temps Réel' **RTC** avec un quartz connecté sur les broches **TOSC1** et **TOSC2 (PC7)** pour servir de source d'horloge indépendante. L'oscillateur est optimisé pour l'utilisation d'un quartz de **32 768 Hz**. L'application d'une source d'horloge externe sur **TOSC1** n'est pas recommandée.

Les choix possibles de pré division sont : **clkT2S/8**, **clkT2S/32**, **clkT2S/64**, **clkT2S/128**, **clkT2S/256** et **clkT2S/1024**. Le choix de la fréquence **clkT2S** ainsi que le 0 (arrêt) sont acceptés. La mise à 1 du bit **PSR2** dans **SFIOR** remet à 0 le pré diviseur. Cela permet à l'utilisateur de fonctionner avec un pré diviseur prévisible.

Le registre **ASSR** sera vu en détail dans le chapitre suivant sur le Timer/Compteur2.

### Registre **SFIOR (Special Function I/O Register)**

Le registre spécial **SFIOR** permet de modifier le fonctionnement du Timer/Compteur2 avec **PSR2**.

Adresse	7	6	5	4	3	2	1	0
\$30	<b>ADTS2</b>	<b>ADTS1</b>	<b>ADTS0</b>	-	<b>ACME</b>	<b>PUD</b>	<b>PSR2</b>	<b>PSR10</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**PSR2 Prescaler Reset Timer/Counter2** Quand ce bit est à 1, le pré diviseur du Timer/Counter2 sera remis à 0. Le bit est remis à 0 par le matériel après l'exécution de l'opération. L'écriture d'un zéro dans ce bit n'aura aucun effet. Ce bit sera toujours lu à zéro si le Timer/Counter2 est cadencé par l'horloge interne de l'UC. Si ce bit est écrit quand le Timer/Counter2 fonctionne dans le mode synchrone, le bit restera à un tant que le pré diviseur n'a pas été remis à 0. Voir le chapitre du Timer/Compteur2.



## Le Timer/Compteur2 à 8 Bits (Timer2)

Le Timer/Compteur2 à usage général à 8 bits avec les particularités principales :

- Simple Compteur à 8 bits
- Comparaison à Rechargement Automatique
- Générateur de Modulation de Phase en Largeur d'Impulsion Correct (PWM)
- Générateur de Fréquence
- Horloge à 10 bits de Pré diviseur
- Source d'Interruption sur Débordement et Comparaison (**TOV2 et OCF2**)
- Horloge Externe 32 768 Hz par Quartz Indépendant

Le synoptique de fonctionnement du Timer/Compteur2 est donné en figure :

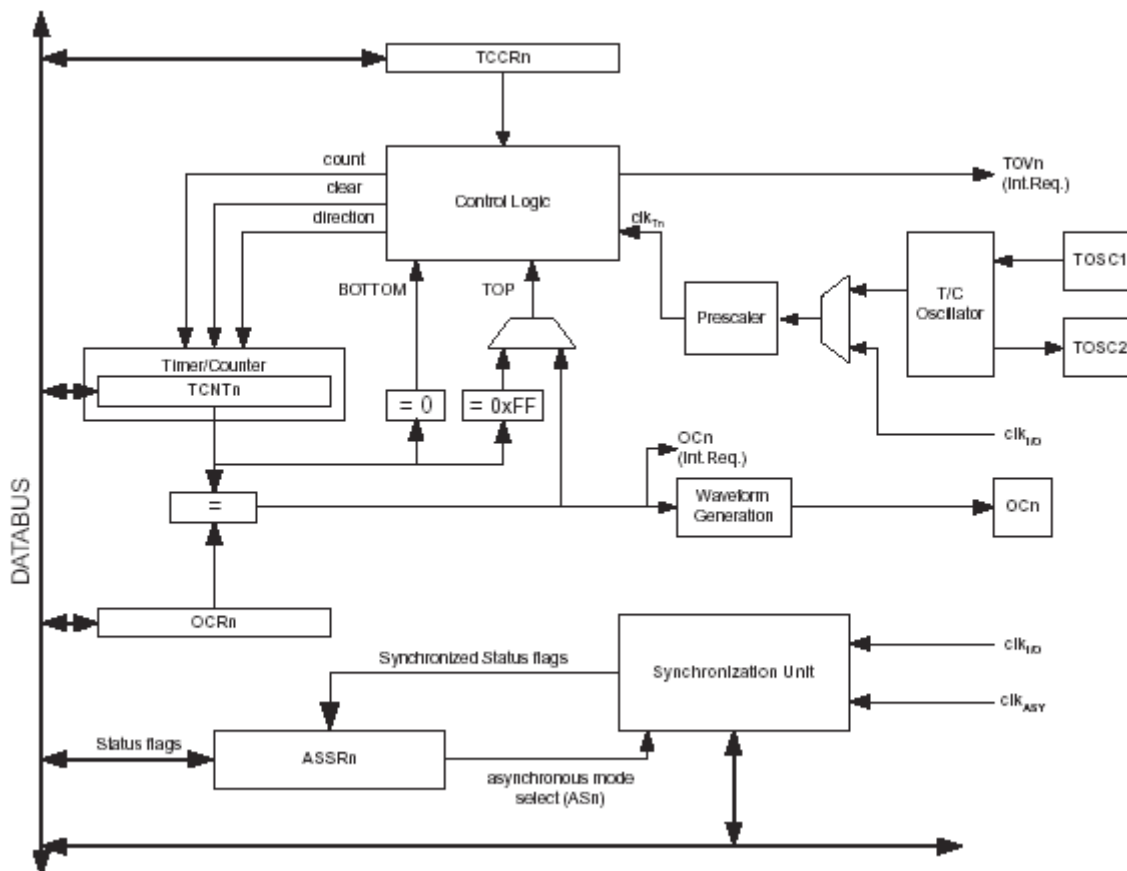


Figure 31, Synoptique du Timer/Compteur2 à 8 bits.

Pour la mise en place des broches d'entrée-sortie, référez-vous au chapitre sur les entrée/sortie numériques (port).

### **Fonctionnement du Timer/Compteur2**

Le compteur du Timer/Compteur2 **TCNT2** est comparé à **OCR2** pour activer des interruptions visibles dans le registre **TIFR** et qui sont masquables dans registre de masque **TIMSK**.

Le Timer/Compteur peut être cadencé intérieurement, via le pré diviseur ou d'une manière asynchrone par la broche **TOSC1/2 (PC6 & 7)**. L'opération asynchrone est contrôlée par le registre de statut asynchrone **ASSR** avec le bit **AS2**. La logique de commande peut bloquer la source d'horloge **clkT2** que le Timer/Compteur2 emploie pour incrémenter (ou décrémenter) sa valeur. Le Timer/Compteur2 est inactif quand aucune source d'horloge n'est choisie. La source d'horloge **clkT2** est par défaut égal à l'horloge système **clkI/O**.

La valeur du comparateur **OCR2** est comparée en permanence avec la valeur du compteur **TCNT2** du Timer/Compteur2. Le résultat de cette comparaison peut être employé pour le générateur de forme d'onde

**PWM** ou pour la production de fréquence variable sur la broche **OC2 (PD7)**. Le drapeau de comparaison **OCF2** placé sur l'égalité de comparaison et peut être employé pour produire une demande d'interruption.

Selon le mode de fonctionnement employé, le Timer/Compteur2 est mis à 0, incrémenté, ou décrémenté à chaque top d'horloge **clkT2**. **ClkT2** peut être produit d'une source d'horloge externe ou interne, choisi par les bits **CS22:20**. Quand aucune source d'horloge n'est choisie **CS22:20** à 0, le Timer/Compteur2 est arrêté. Cependant, la valeur de **TCNT2** peut être lue par l'UC indépendamment.

L'ordre du comptage est déterminé par le bit **WGM21** et le bit **WGM20** du registre de contrôle **TCCR2**.

Si le Timer/Compteur2 déborde, le drapeau **TOV2** sera mis à 1 selon le mode de fonctionnement choisi par les bits **WGM21:20**. **TOV2** peut être employé pour produire une interruption sur débordement.

Le mode comparateur à 8 bits compare continuellement **TCNT2** avec le registre **OCR2**, chaque fois que **TCNT2** est égale à **OCR2**, le drapeau **OCF2** sera mis à 1 au cycle d'horloge suivant. Si la permission d'interruption **OCIE2** est à 1, la demande d'interruption sera générée. Le drapeau **OCF2** est automatiquement remis à 0 quand l'interruption est exécutée. Vous pouvez remettre à 0 le drapeau **OCF2**. Le générateur de forme d'onde peut produire des signaux sur **OC2** en fonction des bits de **WGM21:20** et **COM21:20**.

Les bits **COM21:20** contrôlent les signaux **PWM** qui seront normal ou inversée. Pour des modes non **PWM** le contrôle des bits **COM21:20** détermine si la sortie doit être positionnée à 1, mis à 0, ou utiliser le mode bascule (0 -> 1 ou 1 -> 0) sur une comparaison positive.

## Le mode Normal du Timer2

C'est le mode de fonctionnement le plus simple **WGM21:20** à 0. Dans ce mode la direction du comptage est toujours incrémenté et aucune remise à 0 n'est exécutée. Le compteur déborde simplement quand il dépasse la valeur maximale sur 8 bits soit **\$FF** et reprend ensuite à **\$00** en plaçant le drapeau **TOV2** à 1 au passage à 0. Le drapeau **TOV2** dans ce cas se comporte comme un neuvième bit, sauf qu'il est seulement mis, sans être remis à 0 automatiquement. Cependant, combiné avec l'interruption de débordement cela peut remettre à 0 automatiquement le drapeau **TOV2**. Le registre **TCNT2** peut être modifié à tout moment sans considération spéciale.

## Le mode Comparaison à Remise à 0 (CTC)

Dans le mode compteur à remise à 0 avec **WGM21:20** à 2, **OCR2** est employé pour manipuler la résolution. Le compteur est mis à 0 quand la valeur de **TCNT2** correspond à **OCR2**, donc quand la valeur supérieure du compteur est atteinte. Ce mode permet de contrôler une plus grande plage de fréquence. Il simplifie aussi l'opération de comptage des événements externes. Une interruption peut être produit chaque fois que la valeur supérieure est atteinte par la mise à 1 du drapeau **OCF2**.

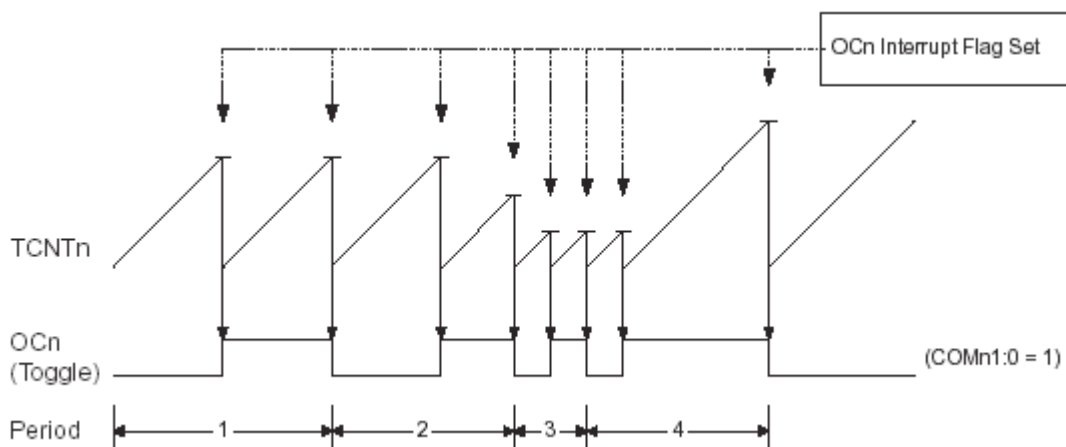


Figure 32, mode CTC.

La valeur **OCR2** peut être changée lors du fonctionnement du compteur si la valeur est supérieure à la valeur du compteur. Si la nouvelle valeur écrite dans **OCR2** est inférieure à la valeur actuelle de **TCNT2**, le comparateur manquera le test et comptera jusqu'au maximum **\$FF** et repartira à **\$00** avant que la

nouvelle comparaison puisse arriver. Pour produire une forme d'onde dans le mode **CTC**, la sortie **OC2** peut être utilisé pour changer son niveau sur chaque comparaison en mettant les bits **COM21:20** à 1. La sortie **OC2** sera visible sur le port quand la direction de données de la broche **OC2** sera placée en sortie. La forme d'onde produite aura une fréquence maximale de  $f_{OC2} = f_{clk\_I/O}/2$  quand **OCR2** est mis à \$00. La fréquence est définie par l'équation suivante :

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

La variable **N** représente le facteur pré diviseur (1, 8, 32, 64, 128, 256, ou 1024). Le drapeau **TOV2** est positionné sur la mise à 0 du compteur du Timer2.

## Le mode PWM Rapide

Le mode de modulation en largeur d'impulsion rapide ou **PWM rapide** défini par **WGM21:20** à 3 fournit une haute fréquence en forme d'onde sur la sortie **OC2**. Le **PWM rapide** diffère de l'autre option **PWM** par son opération sur simple rampe montante. Le compteur part de \$00 et s'incrémente jusqu'à \$FF puis reparte à \$00 et ainsi de suite. Le comparateur met à 1 la sortie **OC2** quand la valeur **OCR2** est la même que **TCNT2**. La sortie **OC2** est mise à 0 lors de la mise à \$00 du compteur. En raison de l'opération sur simple pente montante, la fréquence du mode **PWM rapide** peut être deux fois plus rapide que le mode **PMW correct** (voir chapitre suivant). Le signal sur **OC2** est modifiable par **COM21:20** qui mis à 2 produira un signal **PWM** normal et un signale inversé avec le **COM21:20** à 3.

Dans le mode **PWM rapide** le compteur est incrémenté puis quand la valeur correspondre à la valeur **dOCR2**, le compteur est remis à \$00 au cycle d'horloge suivant. Le drapeau de débordement **TOV2** est mis à 1 chaque fois le compteur atteint le maximum (sur comparaison positive).

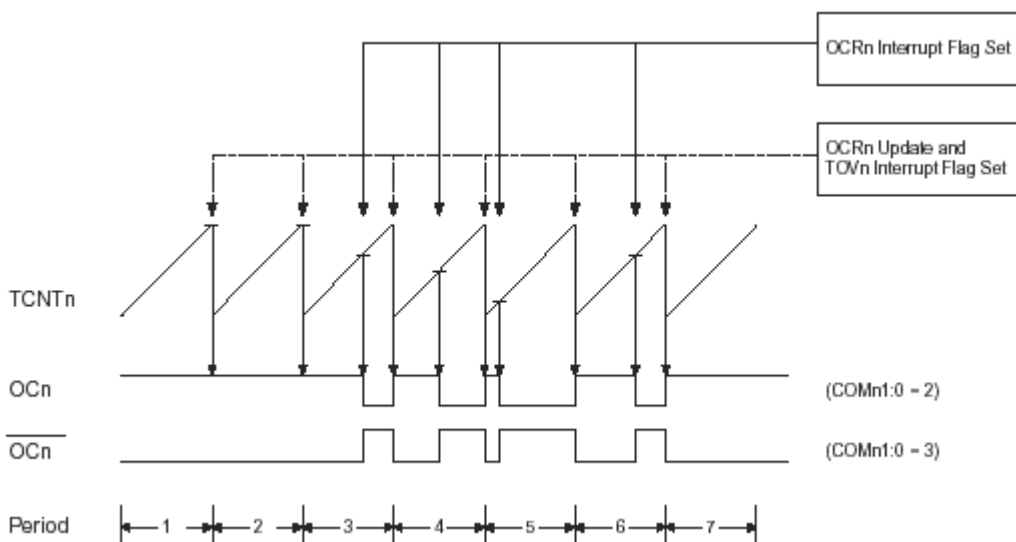


Figure 33, Mode **PWM Rapide**.

La valeur **OC2** est réellement visible sur la broche du port que si la direction de données dans **DDR** est mise en sortie. La fréquence **PWM rapide** peut être calculée par l'équation suivante :

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

La variable **N** représente le facteur pré diviseur (1, 8, 32, 64, 128, 256, ou 1024). Les valeurs extrêmes d'**OCR2** représentent des cas spéciaux. Si **OCR2** est mis à \$00, la production sera une pointe très étroite pour chaque cycle d'horloge. La forme d'onde aura une fréquence maximale de  $f_{OC2} = f_{clk\_I/O}/2$  quand **OCR2** est mis à 0.

## Le Mode PWM Correct (PWM C)

Le mode **PWM correct** définie par **WGM21:20** mis à 1 est basée sur une opération à double pente. Le compteur est inversé à plusieurs reprises. Dans le sens montant la comparaison met à 0 la sortie **OC2**

l'égalité **TCNT2** et **OCR2**, puis dans après l'inversion du compteur sur l'arrivé au sommet, la nouvelle comparaison met à 1 la sortie **OC2** et l'on recommence sur la deuxième pente. L'opération sur une double pente à une fréquence maximale plus basse que l'opération à simple pente.

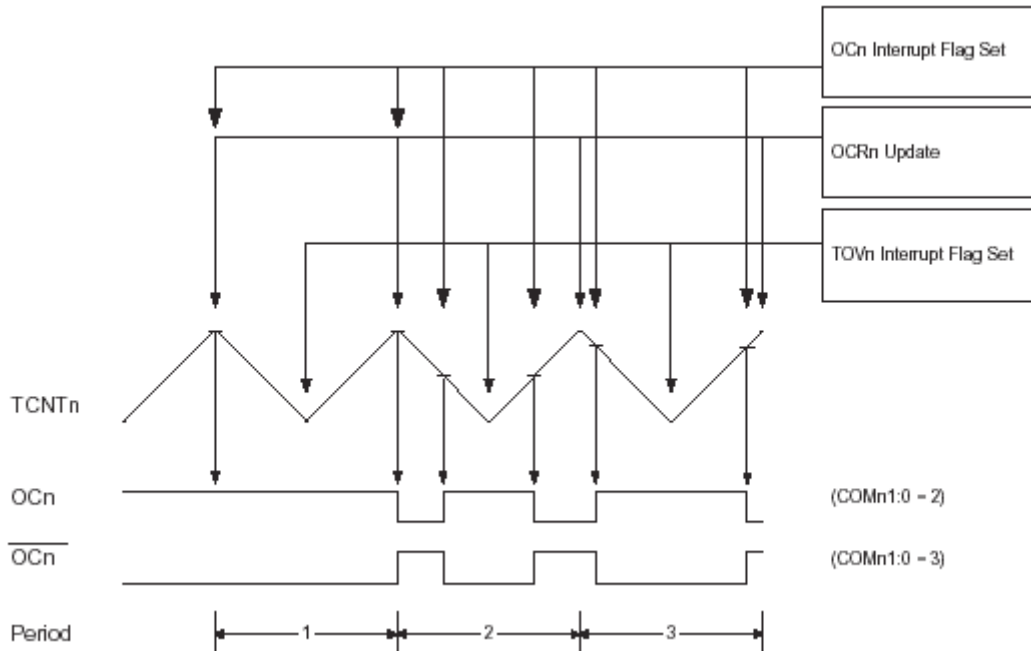


Figure 34, Mode **PWM** Correct.

La résolution est fixée à 8 bits. Dans la phase corrigée, le compteur est incrémenté jusqu'à la valeur maximale avant d'être inversé et le compteur **TCNT2** sera égal au **Max** pour un cycle d'horloge complet du Timer2. Les petites marques horizontales sur les pentes de **TCNT2** représentent les comparaisons entre **OCR2** et **TCNT2**. Le drapeau de débordement **TOV2** est mis à 1 à chaque fois que le compteur atteint \$00.

La génération de formes d'onde **PWM** sur **OC2** est modifiable par **COM21:0** à 2 qui produira un signal **PWM** normal et un signal PWM inversé en mettant **COM21:20** à 3. La sortie **OC2** sera réellement visible sur la broche du port si la direction des données est mis en sortie dans **DDR**. La fréquence **PWM** peut être calculée par l'équation suivante :

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

La variable **N** représente le facteur pré division (1, 8, 32, 64, 128, 256, ou 1024). Les valeurs extrêmes **OCR2** représentent des cas spéciaux, si **OCR2** est mis à \$00, le signal sera toujours en bas et s'il est mis à \$FF le signale sera continuellement en haut.

Au début de la période 2, **OC2** a une transition de haut à bas bien qu'il n'y ait pas comparaison. Le point de cette transition doit garantir la symétrie autour du \$00. IL y a deux cas qui donnent une transition sans comparaison :

- **OCR2** est chargé à la valeur de Max. Quand la valeur **OCR2** est égal à Max la valeur **OC2** est toujours en haut pour assurer la symétrie autour du \$00 la valeur **OC2** est mis en bas.
- le compteur commence à compter avec une valeur plus grande que **d'OCR2** et manque la comparaison, c'est pourquoi **OC2** est modifié sur la montée.

## Le mode Asynchrone du Timer2

Quand Timer2 fonctionne en mode asynchrone, quelques considérations doivent être prises en comptes.

**Avvertissement :** En commutant le Timer2 entre le cadencement asynchrone et synchrone, les registres **TCNT2**, **OCR2** et **TCCR2** pourrait être corrompu. Une procédure sûre pour la commutation de la source d'horloge est la suivante :

1. Mettre hors service les interruptions du Timer2 **OCIE2** et **TOIE2** = 0.

2. Choisir la source d'horloge en mettant **AS2** comme vous le souhaitez.
3. Écrire de nouvelle valeur dans **TCNT2**, **OCR2** et **TCCR2**.
4. Tester l'opération asynchrone : Attendez **TCN2UB**, **OCR2UB** et **TCR2UB**.
5. Placer les interruptions du Timer2.
6. Permettre les interruptions, si nécessaire.

L'oscillateur est optimisé pour l'utilisation d'un quartz de **32 768 Hz**. L'application d'une horloge externe autre sur la broche **TOSC1** n'est pas recommandée. La fréquence d'horloge principale de l'UC doit être quatre fois grande que la fréquence de l'oscillateur ( $4 \times 32768 = 132 \text{ KHz}$  mini).

En écrivant à un des registres **TCNT2**, **OCR2**, ou **TCCR2**, la valeur est transférée dans un registre provisoire et l'accès est fermé pendant deux front positifs de l'horloge **TOSC1** d'où la consultation d'état d'occupation de ces registres.

L'utilisateur ne doit pas écrire une nouvelle valeur avant que le contenu du registre provisoire n'ait été transféré à sa destination. Chacun des trois registres a un registre provisoire individuel, ce qui signifie par exemple que l'écriture dans **TCNT2** ne dérangera pas **OCR2** d'écrire dans le sien. Pour détecter qu'un transfert au registre de destination a eu lieu, le registre de statut asynchrone **ASSR** a été mis en oeuvre.

En entrant au mode sommeil simple ou prolongé après avoir écrit dans **TCNT2**, **OCR2**, ou **TCCR2**, l'utilisateur doit attendre, avant, que le registre écrit soit mis à jour, si le Timer2 est employé pour réveiller le système. Autrement, le **MCU** entrera au mode de sommeil avant que les changements ne soient efficaces. C'est particulièrement important car le **MCU** ne recevra jamais d'interruption pour ce réveiller.

Si le Timer2 est employé pour réveiller le dispositif en mode sommeil, les précautions doivent être prises, l'interruption a besoin d'un cycle **TOSC1** de remise en route. Si le temps de la commande de mise en mode de sommeil est inférieur à 1 cycle **TOSC1**, l'interruption n'arrivera pas à réveiller le système. L'utilisateur dans le doute peut utiliser l'algorithme suivant pour assurer qu'un cycle **TOSC1** s'est écoulé :

1. Écrire une valeur dans **TCCR2**, **TCNT2**, ou **OCR2**.
2. Attendre la fin de la mise à jour par le passage à 0 du drapeau occupé dans **ASSR**.
3. Entrer en mode sommeil.

Quand l'opération asynchrone est choisie, l'oscillateur de **32 768 Hz** du Timer2 court toujours, sauf dans le cas d'une coupure de courant. Après un **RESET** ou la mise en route du **MCU**, l'utilisateur doit être conscient du fait que l'oscillateur peut prendre plus d'une seconde pour ce stabiliser.

Le contenu de tous les registres du Timer2 doit être considéré comme perdu après une coupure de courant en raison de l'instabilité du signal d'horloge pendant le démarrage.

Lors du réveille du mode sommeil quand le compteur est cadencé d'une manière asynchrone : le compteur débute sur le cycle suivant de l'horloge du Timer2, c'est-à-dire le compteur est toujours décaler d'au moins un cycle avant que le processeur ne puisse lire la valeur.

Après la commande, le **MCU** est interrompu pour quatre cycles, il exécute la routine d'interruption et reprend l'exécution de l'instruction après l'appel au mode sommeil.

La lecture du registre **TCNT2** peu de temps après la commande de mise en sommeil peut donner un résultat incorrect. **TCNT2** est cadencé sur l'horloge asynchrone **TOSC**, la lecture de **TCNT2** doit être fait par un registre synchronisé sur l'horloge d'entrée-sortie interne. La synchronisation a lieu pour chaque front montant de **TOSC1**. La procédure recommandée pour la lecture de **TCNT2** est :

1. Écrire n'importe quelle valeur dans **OCR2** ou **TCCR2**.
2. Attendre que la mise à jour par la présence d'un 0 dans le drapeau correspondant.
3. Lire **TCNT2**.

Pendant l'opération asynchrone, la synchronisation des drapeaux d'interruption pour le compteur asynchrone prend trois cycles de processeur plus un cycle du Timer2. La comparaison en sortie est synchronisée sur l'horloge de Timer2 et non sur l'horloge de processeur.

## Les Registres Associés au Timer/Compteur2

Le Timer2 utilise 4 registres spécifiques et deux registres d'interruption.

## Registre TCCR2 (Timer/Counter2 Control Register A)

Le registre de control du Timer/Compteur2.

Adresse	7	6	5	4	3	2	1	0
\$25	<b>FOC2</b>	<b>WGM20</b>	<b>COM21</b>	<b>COM20</b>	<b>WGM21</b>	<b>CS22</b>	<b>CS21</b>	<b>CS20</b>
L/E	L/E	L/E	L/E	L/E	E	E	L/E	L/E

**FOC2 Force Output Compare** Ce bit est actif seulement quand un mode non **PWM** est sélectionné. Cependant, pour assurer la compatibilité avec des dispositifs futurs, ce bit doit être mis à 0 quand **TCCR2** est écrit dans un mode **PWM**. En écrivant un 1 dans **FOC2**, l'état de la comparaison est imposée à l'unité de Génération de Forme d'Onde et l'état de la sortie **OC2** est changée selon les bits **COM21:20**. C'est une fonction de type **strobe**. Un **FOC2 strobe** ne produira pas d'interruption, donc il ne mettra pas à 0 le compteur dans le mode **CTC** avec **OCR2** comme sommet. Le bit **FOC2** est toujours lu à 0.

**WGM20 Waveform Generation Mode** Voir le bit **WGM21** ci-dessous

**COM21&20 Compare Output Mode 21 & 20** Ces bits contrôlent la production de comparaison sur la sortie **OC2**. Si un ou les deux bits sont écrits à 1, la sortie **OC2** ignore la fonctionnalité normale d'entrée-sortie à laquelle il est connecté. Cependant, notez que le registre de direction des données **DDR** doit être placé en sortie. Le tableau qui suit précise le fonctionnement des sorties **OC2** en fonction du mode choisi dans **WGM21:20** qui doit être **Normal** ou **CTC**, mais **pas de mode PWM**.

<b>COM21</b>	<b>COM20</b>	<b>Description</b>
0	0	Opération de port Normale, <b>OC2</b> débranché
0	1	Interrupteur à bascule <b>OC2</b> sur comparaison
1	0	Mis à 0 d' <b>OC2</b> sur comparaison
1	1	Mis à 1 d' <b>OC2</b> sur comparaison

Dans le mode **PWM rapide uniquement** :

<b>COM21</b>	<b>COM20</b>	<b>Description</b>
0	0	Opération de port Normal, <b>OC2</b> débranché.
0	1	<i>Réservé</i>
1	0	Mis à 0 d' <b>OC2</b> sur comparaison, mis à 1 à \$FF
1	1	Mis à 1 d' <b>OC2</b> sur comparaison, mise à 0 à \$FF

Dans le mode **PWM correct uniquement** :

<b>COM21</b>	<b>COM20</b>	<b>Description</b>
0	0	Opération de port Normale, <b>OC2</b> débranché.
0	1	<i>Réservé</i>
1	0	<b>OC2</b> Mis à 0 sur comparaison en montant. <b>OC2</b> Mis à 1 sur comparaison en descendant.
1	1	<b>OC2</b> Mis à 1 sur comparaison en montant. <b>OC2</b> Mis à 0 sur comparaison en descendant.

**WGM11&10 Waveform Generation Mode** Ces bits déterminent le fonctionnement du Timer/Compteur2 :

<b>Mode</b>	<b>WGM21 (CTC2)</b>	<b>WGM20 (PWM2)</b>	<b>Mode d'Opération</b>	<b>Mise à jour d'OCR2</b>	<b>Active TOV2 sur</b>
0	0	0	Normal	Immediate	\$FF
1	0	1	PWM Correct	\$FF	\$00
2	1	0	CTC	Immediate	OCR2
3	1	1	PWM Rapide	\$FF	\$FF

**CS22:20 Clock Select** Les trois bits de choix de l'horloge permettent de sélectionner la source d'horloge employé par le Timer/Compteur2 :

CS02	CS01	CS00	Source
0	0	0	0 (stop le compteur)
0	0	1	Horloge système / 1
0	1	0	Horloge système / 8
0	1	1	Horloge système / 32
1	0	0	Horloge système / 64
1	0	1	Horloge système / 128
1	1	0	Horloge système / 256
1	1	1	Horloge système / 1024

### Registre TCNT2 (Timer/Counter2 Register)

Le registre **TCNT2** donne l'accès direct au compteur à 8 bits du Timer2. La modification du compteur **TCNT2** tandis que le compteur court présente un risque de manquer une comparaison entre **TCNT2** et **OCR2**.

Adresse	7	6	5	4	3	2	1	0
\$24	<b>TCNT27</b>	<b>TCNT26</b>	<b>TCNT25</b>	<b>TCNT24</b>	<b>TCNT23</b>	<b>TCNT22</b>	<b>TCNT21</b>	<b>TCNT20</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**TCNT2 Timer/Counter2 Register** C'est le contenu du compteur du Timer2 sur 8 bits.

### Registre OCR2 (Output Compare Register 2)

Le registre **OCR2** donne l'accès direct au comparateur à 8 bits du Timer2.

Adresse	7	6	5	4	3	2	1	0
\$23	<b>OCR27</b>	<b>OCR26</b>	<b>OCR25</b>	<b>OCR24</b>	<b>OCR23</b>	<b>OCR22</b>	<b>OCR21</b>	<b>OCR20</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCR2 Output Compare Register 2** C'est le contenu du comparateur du Timer2 sur 8 bits qui est continuellement comparée avec la valeur **TCNT2** et produit de forme d'onde sur la broche **OC2** lors de l'égalité.

### Registre ASSR (Asynchronous Status Register)

Le registre permet de sélectionner le mode **RTC**, soit l'utilisation du quartz connecté sur **TOSC1/2** pour créer une horloge basé sur le temps calendaire (à la seconde).

Si une écriture est exécutée sur l'un des trois registres du Timer2 précédant pendant sa mise à jour et que le drapeau occupé est mis, la valeur mise à jour pourrait être perdue et causer une interruption involontaire. Les mécanismes pour la lecture de **TCNT2**, **OCR2** et **TCCR2** sont différents. En lisant **TCNT2**, la valeur de compteur est réellement lue. En lisant **OCR2** ou **TCCR2**, la valeur du registre de stockage provisoire est lue.

Adresse	7	6	5	4	3	2	1	0
\$22	-	-	-	-	<b>AS2</b>	<b>TCN2UB</b>	<b>OCR2UB</b>	<b>TCR2UB</b>
L/E	L	L	L	L	L/E	L	L	L

**AS2 Asynchronous Timer/Counter2** C'est le mode Asynchrone du Timer2. Quand **AS2** est écrit à 0, le Timer2 est cadencé par l'horloge d'entrée-sortie, **clkI/O**. Quand **AS2** est écrit à 1, le Timer2 est cadencé par l'oscillateur à quartz connecté sur **TOSC1/2**. Quand la valeur d'**AS2** est changée, le contenu de **TCNT2**, **OCR2** et **TCCR2** n'est pas fiable, il doit être initialisé en fonction de vos paramètres.

**TCN2UB Timer/Counter2 Update Busy** Quand Timer2 fonctionne en mode asynchrone **AS2** = 1 et **TCNT2** est écrit, ce bit est mis à 1. Quand **TCNT2** a été mis à jour du registre de stockage provisoire, ce bit est mis à 0 par le matériel. Un 0 dans ce bit indique que **TCNT2** est prêt à être mis à jour avec une nouvelle valeur.

**OCR2UB Output Compare Register2 Update Busy** Quand Timer2 fonctionne en mode asynchrone et **OCR2** est écrit, ce bit est mis à 1. Quand **OCR2** a été mis à jour du registre de stockage provisoire, ce bit est mis à 0 par le matériel. Un 0 dans ce bit indique **qu'OCR2** est prêt à être mis à jour avec une nouvelle valeur.

**TCR2UB Timer/Counter Control Register2 Update Busy** Quand Timer/Counter2 fonctionne en mode asynchrone et **TCCR2** est écrit, ce bit est mis à 1. Quand **TCCR2** a été mis à jour du registre de stockage provisoire, ce bit est mis à 0 par le matériel. Un 0 dans ce bit indique que **TCCR2** est prêt à être mis à jour avec une nouvelle valeur.

### Registre TIFR (Timer/Counter Interrupt Flag)

Le registre **TIFR** indique l'état des interruptions internes des modules Timer et Comparateur. Les interruptions seront prises en charge si le bit correspondant dans le registre **TIMSK** à été activé (mis à 1). Ce registre sera revu dans les chapitres consacrés aux Timer.

Adresse	7	6	5	4	3	2	1	0
\$38	<b>OCT2</b>	<b>TOV2</b>	<b>ICF1</b>	<b>OCF1A</b>	<b>OCF1B</b>	<b>TOV1</b>	<b>OCF0</b>	<b>TOV0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCF2 Output Compare Flag 2** Le bit est mis à 1 quand une donnée est arrivée et comparée entre le Timer/Counter2 et le contenu du registre d'**OCR2**. Le bit **OCF2** est remis à 0 par le matériel en exécutant le vecteur de l'interruption. Vous pouvez aussi forcer **OCF2** à 0.

**TOV2 Timer/Counter2 Overflow Flag** Le bit est mis à 1 quand le Timer/Counter2 déborde. **TOV2** est mis à 0 par le matériel lors de l'exécution de l'interruption. Vous pouvez aussi forcer **TOV2** à 0. Dans le mode **PWM**, ce bit est mis à 1 quand le Timer/Counter2 arrive à la valeur \$00.

### Registre TIMSK (Timer/Counter Interrupt Mask)

Le registre **TIMSK** définit individuellement le masque pour les modules Timer, Compteurs et Comparateurs. Le masque autorise les interruptions s'il est à 1, s'il est à 0 l'interruption ne sera pas prise en compte. Lors de l'interruption, le registre **TIFR** contiendra le bit du module qui à activer l'interruption et qu'il faudra tester pour connaître le modules en cause. Ce registre sera revu dans les chapitres consacrés aux Timer.

Adresse	7	6	5	4	3	2	1	0
\$39	<b>OCIE2</b>	<b>TOIE2</b>	<b>TICIE1</b>	<b>OCIE1A</b>	<b>OCIE1B</b>	<b>TOIE1</b>	<b>OCIE0</b>	<b>TOIE0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**OCIE2 Timer/Counter2 Output Compare Match Interrupt Enable** Quand le bit est à 1 l'interruption est validée pour le Timer/Counter2 et le bit **OCF2** est modifié dans le registre **TIFR**.

**TOIE2 Timer/Counter2 Overflow Interrupt Enable** Quand le bit est à 1 l'interruption est validée pour le Timer/Counter2 et le bit **TOV2** est modifié dans le registre **TIFR**.

## Programmation du Timer/Compteur2

### Paramétrage du TIMER2

Si interruption penser à mettre l'instruction **SEI**

Dans le même cas, mettre à 1 le drapeau **TOIE2** du registre **TIMSK**

Inscription de la valeur du compteur avec **TCNT2**

Sélection de la source d'horloge du compteur dans **TCCR2**, qui met le compteur en fonctionnement

Exécution de la routine d'interruption et rechargement du compteur.

### Exemple 1 Timer2 mode asynchrone

L'exemple présente la façon d'employer le Timer2 en mode interruption sur comparaison. Le compteur sera configuré pour que l'événement de comparaison arrive chaque seconde. Cette particularité pourrait être employée pour mettre en oeuvre une horloge temps réel type **RTC**. Dans cet exemple, cependant, les broches du port seront inversées à chaque événement de comparaison pour faire clignoter une **LED** avec une fréquence de 0,5 **Hz**.



Le Port **B** doit être connecté aux **LEDs** et le Port **D** à l'interface de programmation type **STK500**. De plus, le quartz de 32 768 **Hz** doit être monté sur les broches **TOSC1/PC6** et **TOSC2/PC7** du Port **C**.

La configuration du Timer2 peut être calculée selon l'équation ci dessous. La valeur maximum du compteur **MaxVal** sera mise dans **OCR2** pour la comparaison. La pré division de l'horloge **PCKx** est dans ce cas le signal d'horloge du cristal **fOSCCK**, et **TOVCK** le signal d'horloge pour les événements de changement d'état du Port **B** à 1 seconde. L'équation suivante montre à la description mathématique de cette relation :

$$1 = TOV_{CK} = \frac{f_{OSCCK}}{PVal \cdot OCR2} = \frac{32.768 \text{ kHz}}{PVal \cdot OCR2}$$

Une valeur de pré-division de 1024 est choisie et **OCR2** est mis à 32 pour obtenir un temps d'une seconde entre deux événements de comparaison.

Le programme d'initialisation suivant montre comment fonder un tel système :

```

init_Ex1:    ldi    r16,1<<AS2           ; Activation en mode asynchrone
             out    ASSR,r16         ; Active le mode asynchrone, timer=0 sur comparaison
             ldi    r16,(1<<CTC2)|(1<<CS22)|(1<<CS21)|(1<<CS20)
             out    TCCR2,r16        ; Horloge Timer = horloge système / 1024
             ldi    r16,1<<OCF2
             out    TIFR,r16         ; OCF2 = 0 pendant l'interruption
             ldi    r16,1<<OCIE2
             out    TIMSK,r16        ; Active Timer sur interruption sur comparaison
             ldi    r16,32
             out    OCR2,r16         ; Compare sur la valeur 32
             ser    r16
             out    DDRB,r16         ; Init. Port D en sortie
loop:        sbic   ASSR, OCR2UB      ; Attendre la mise à jour des registres
             rjmp  loop
             ret

```

Dans le programme suivant la routine de service d'interruption doit être mise en oeuvre. Cette routine sera exécutée avec chaque événement de comparaison. Le but est de basculer les bits du Port B (les **LEDs**).

```

ISR_OCIE2:  push   r16
             In    r16,SREG
             Push  r16
             In    r16,PORTB         ; Lire Port B
             Com   r16               ; Inversion des bits de r16
             Out   PORTB,r16         ; Ecrire Port B
             Pop   r16
             Out   SREG,r16
             Pop   r16
             reti

```

### Exemple 2 : Timer2 mode PWM à 8 bits

Cet exemple montre comment produire des ondes sur la broche **PD7/OC2**. Pour observer cela, le Port **D7** doit être connecté à une **LED**. L'initialisation de la production du signal **PWM** aura une relation de 1/8 à 7/8 (**OCR2 = 0xE0**) d'horloge sur un cycle.

Le routine d'initialisation suivant montre comment fonder un tel système :

```

init_Ex2:                                       ; 8 bit PWM non-inversé (Fck/510)
             ldi    r16,(1<<PWM2)|(1<<COM21)|(1<<CS20)
             out    TCCR2,r16           ; 8 bit PWM non-inversé (Fck/510)
             ldi    r16,0xE0
             out    OCR2,r16           ; Ration du comparateur
             ldi    r16,0x8F
             out    DDRD,r16           ; active PD7/OC2 et le Port D en sortie
             ret

```

# L'interface Série Synchrones SPI

L'interface SPI est l'abréviation *Serial Peripheral Interfacel Synchronous*, soit Interface Série Synchrones. Contrairement à l'UART et comme son nom l'indique, ce type de périphérique génère les signaux d'horloge de synchronisation, arbitré par un Maître, c'est une interface série à 4 fils.

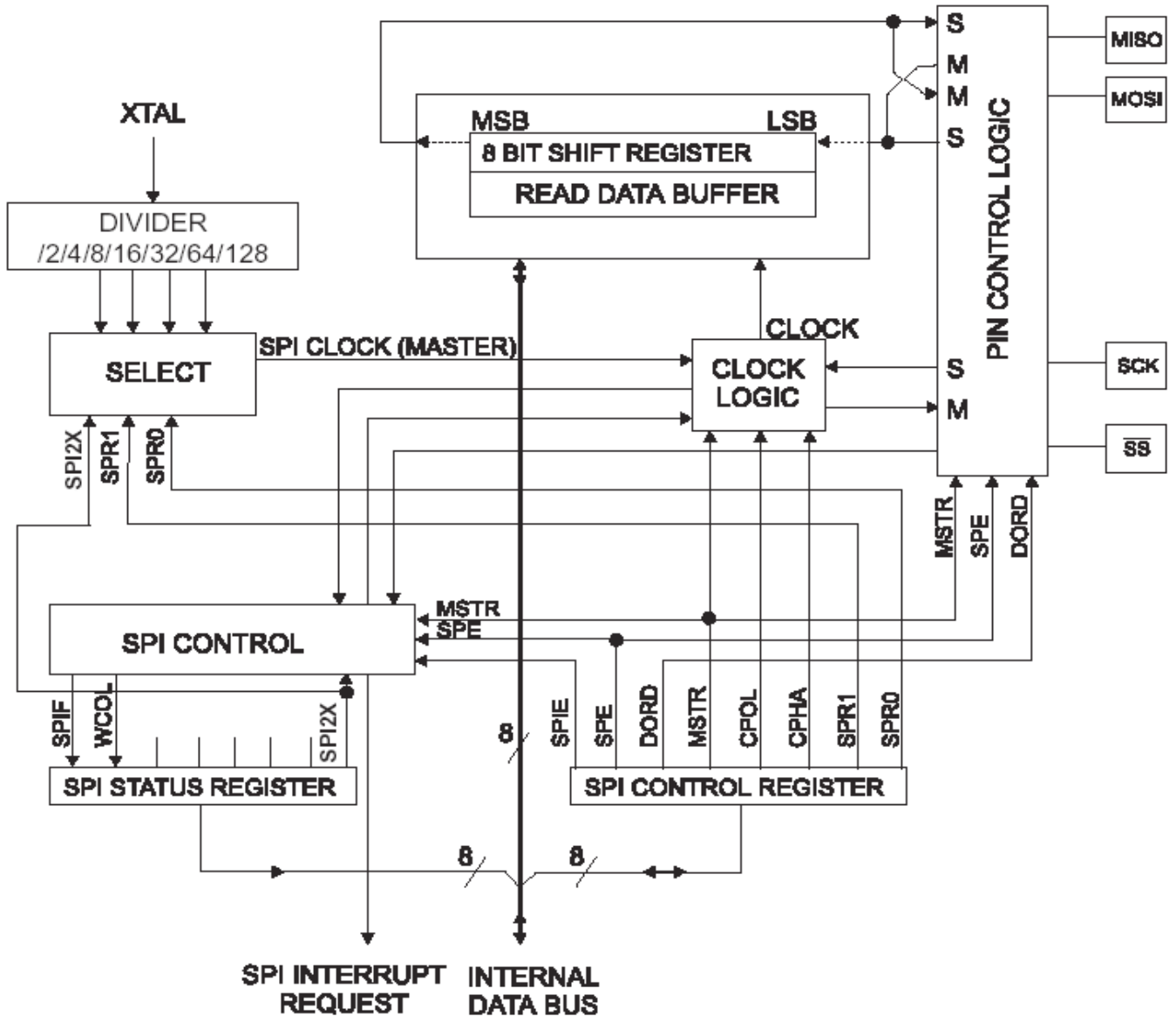


Figure 35, Synoptique de l'interface SPI.

L'interface SPI permet le transfert de données ultrarapide synchrone entre l'ATMEGA et des périphériques ou entre plusieurs dispositifs. L'interface SPI inclut les particularités suivantes :

- Transfert de Données Full Duplex à Quatre Fils Synchrone
- Maître ou Esclave
- Transfert de Données avec **LSB** d'abord ou **MSB** en premier
- 7 Taux de Transfert Programmables
- Drapeau de Fin de Transmission pour Interruption
- Protection de Drapeau de Collision
- Mode Inoccupé
- SPI** Mode Maître à Vitesse Double ( $CK/2$ )

## Description des Broches

Le **SPI** comporte quatre broches, appeler ligne de communication :

**MISO** : *Master In Slave Out* entrée des données série sur le Maître, sorties sur l'Esclave

**MOSI** : *Master Out Slave* sortie des données série pour le Maître, entrée pour l'esclavage

**SCK** : *Serial Clock* signaux d'horloge de synchronisation des circuits connectés sur le Bus. Ces signaux d'horloges sont générés par le Maître.

**SS** : *Slave Select* sélection du mode Maître de la liaison, le Maître la met à l'état haut.

Le tableau qui suit donne la configuration des broches par rapport au mode de fonctionnement :

Broche	Direction Maître SPI	Direction Esclave SPI
<b>MOSI</b>	Défini par l'utilisateur	Entrée
<b>MISO</b>	Entrée	Défini par l'utilisateur
<b>SCK</b>	Défini par l'utilisateur	Entrée
<b>SS</b>	Défini par l'utilisateur	Entrée

La connexion des interfaces sera la suivante :

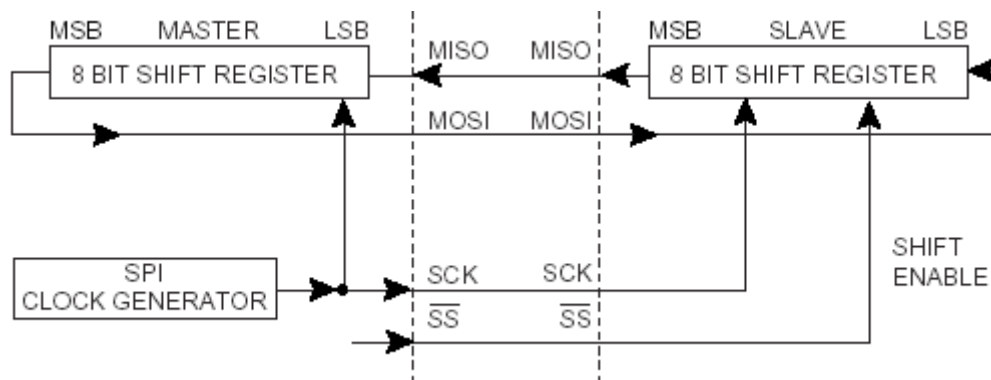


Figure 36, Raccordement entre microcontrôleurs

## Le fonctionnement de l'interface SPI

Ce type d'interface nécessite de définir un Maître et un Esclave en fonction de l'état de la broche '/SS'.

Le système est constitué de deux registres à décalage et d'un générateur d'horloge. Le **SPI** Maître amorce le cycle de communication en mettant en bas (0) la ligne **SS** de l'Esclave désiré. Le Maître et l'Esclave préparent les données à être envoyés dans leurs registres à décalage respectifs et le Maître produit les impulsions d'horloge exigées sur la ligne **SCK** pour échanger des données. Les données sont toujours transmises du Maître vers l'Esclave par la ligne **MOSI**, et de l'Esclave au Maître sur la ligne **MISO**. Après chaque paquet de données, le Maître synchronisera l'Esclave en tirant en haut la ligne **SS**.

Quand l'interface **SPI** est configurée en Maître, il n'y a aucun contrôle automatique de la ligne **SS**. Cela doit être traité par le logiciel d'utilisateur avant que la communication ne puisse commencer, le programme doit mettre en bas la ligne **SS** et quand c'est fait, l'écriture d'un octet dans le registre de données permet à l'interface **SPI** de commencer à générer l'horloge **SCK** et le matériel transfère les huit bits dans l'Esclave. Après le chargement d'un octet, l'horloge s'arrête et le drapeau de fin de transmission **SPIF** est placé à l'état haut. Si l'interruption est autorisée **SPIE=1** dans le Registre **SPCR**, une interruption est demandée. Le Maître peut continuer à charger l'octet suivant en l'écrivant dans **SPDR**, ou en signalant la fin du paquet en plaçant la ligne **SS** en haut. Le dernier octet entrant sera maintenu dans le registre de réception pour une utilisation ultérieure.

Quand l'interface **SPI** est configurée en Esclave, elle restera en sommeil, avec **MISO** en mode trois états, tant que la ligne **SS** reste à l'état haut. Dans cet état, le logiciel peut mettre à jour le contenu du registre de données **SPDR**, mais les données ne seront pas chargées. Dès que **SS** est mis à l'état bas, les signaux d'horloge **SCK** assurant le transfert de l'octet seront reçus. Quand un octet a été complètement chargé, le drapeau de fin de transmission **SPIF** sera mis à 1. Si l'interruption **SPI** est validée **SPIE**, une interruption est demandée. L'Esclave peut continuer à placer de nouvelles données à être envoyées dans **SPDR** avant la

lecture des données entrantes. Le dernier octet entrant sera maintenu dans le registre de transmission pour une utilisation ultérieure.

Le système est à simple buffer en transmission et double buffer en réception. Cela signifie que les octets à transmettre ne peuvent pas être écrits dans le registre de données du SPI avant que le cycle de chargement entier ne soit achevé. En réception de donnée, un octet reçu doit être lu dans le registre de données du SPI avant que l'octet suivant n'ait été complètement chargé dans, autrement, le premier octet sera perdu.

Dans le mode Esclave, l'échantillonnage du signal entrant est synchronisé sur la broche **SCK**. Pour assurer le prélèvement correct d'échantillons du signal d'horloge, la fréquence de l'horloge SPI ne doit jamais excéder  $F_{osc}/4$ .

L'interface SPI est aussi utilisée par le système pour la programmation de la mémoire **FLASH** et de l'**EEPROM**.

## Fonctionnalité de la broche SS

### Le mode d'Esclave

Quand le SPI est configuré en mode Esclave, la broche **SS** est toujours en entrée. Quand **SS** est à l'état bas, le SPI est activé et **MISO** transfère les données si c'est la configuration définie par l'utilisateur. Toutes les autres broches sont des entrées. Quand **SS** est à l'état haut, toutes les broches sont des entrées et le SPI est passif, ce qui signifie qu'il ne recevra pas de données entrantes. Notez que le SPI sera remis à 0 quand **SS** passe à l'état haut. La broche **SS** est utile pour la synchronisation de paquet/octet pour tenir le compte de bit et garder l'Esclave synchrone avec le générateur d'horloge Maître.

### Le mode de Maître

Quand le SPI est configuré en Maître (**MSTR=1** dans **SPCR**), l'utilisateur peut déterminer la direction de la broche **SS**.

Si **SS** est configuré en sortie, la broche **SS** pilotera l'Esclave du SPI.

Si **SS** est configuré comme une entrée, elle doit être à l'état haut pour assurer le fonctionnement normal du Maître. Si on met **SS** à l'état bas quand le SPI est configuré en Maître, le système interprètera cela comme un autre Maître qui choisit le SPI comme un Esclave et commencera à y envoyer des données. Pour éviter ce problème, le système SPI prend les actions suivantes :

1. Le bit **MSTR** dans **SPCR** est mis à 0 et le système SPI devient un Esclave, **MOSI** et **SCK** deviennent des entrées.
2. Le drapeau **SPIF** dans **SPSR** est mis à 1 et si les interruptions sont en services, la routine d'interruption sera exécutée.

Ainsi, quand la transmission SPI est interrompue en mode Maître, il existe là une possibilité que **SS** soit eu niveau bas, l'interrompant doit toujours vérifier que le bit **MSTR** soit toujours mis à 1. Si le bit **MSTR** a été mis à 0 par un Esclave, il doit être remis à 1 par l'utilisateur pour permettre à nouveau le mode de Maître.

## Modes de Données

Il y a quatre combinaisons de phase d'horloge **SCK** et de polarité, en ce qui concerne des données, qui sont déterminés par les bits de contrôle **CPHA** et **CPOL** du registre **SPCR**. On montre les formats de transfert de données SPI dans les figures qui suivent. Les bits de données sont chargés et échantillonnés sur les fronts opposés du signal **SCK**, assurant le temps suffisant pour stabiliser les signaux de données.

<b>CPOL</b>	<b>CPHA</b>	<b>Front Principal</b>	<b>Front Final</b>	<b>SPI Mode</b>
0	0	Échantillon (Montant)	Initialisation (Tombant)	0
0	1	Initialisation (Montant)	Échantillon (Tombant)	1
1	0	Échantillon (Tombant)	Initialisation (Montant)	2
1	1	Initialisation (Tombant)	Échantillon (Montant)	3

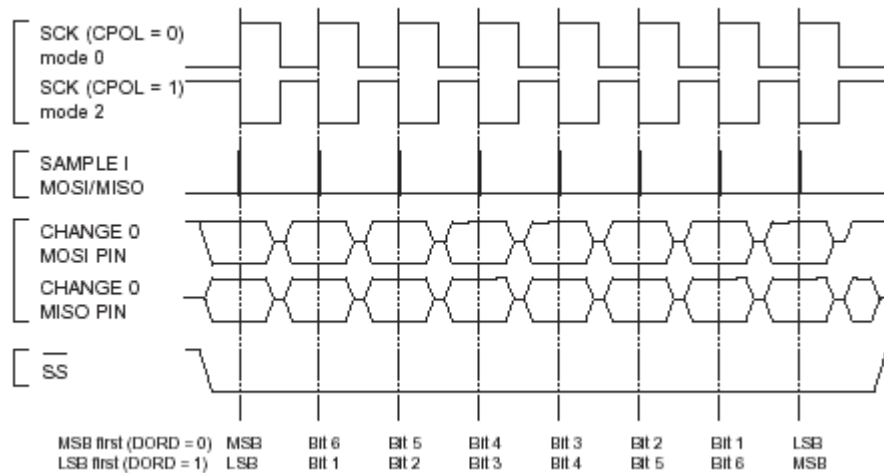


Figure 37, Chronogramme des signaux de l'interface avec **CPHA=0**.

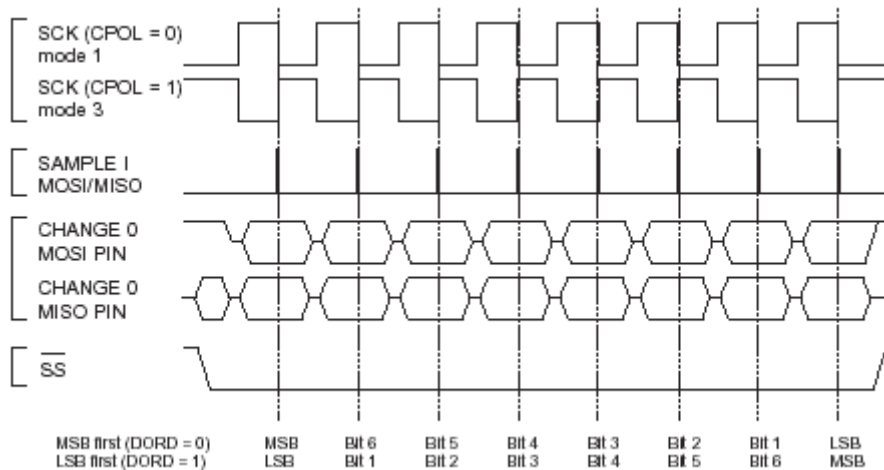


Figure 38, Chronogramme des signaux de l'interface avec **CPHA=1**.

## Les registres de l'interface SPI

Ils sont au nombre de 3 et permettent de contrôler l'interface **SPI**.

### Registre SPCR (*SPI Control Register*)

Le registre de contrôle de l'interface **SPI**.

Adresse	7	6	5	4	3	2	1	0
\$0D	<b>SPIE</b>	<b>SPE</b>	<b>DORD</b>	<b>MSTR</b>	<b>CPOL</b>	<b>CPHA</b>	<b>SPR1</b>	<b>SPR0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**SPIE SPI Interrupt Enable** Validation d'interruption **SPI** lors du passage à 1 du bit **SPIF** du registre

**SPE SPI Enable** Mise en marche de l'interface **SPI**

**DORD Data Order** Choix de transmission de l'ordre des bits : Si à 1, le bit de poids faible est émis en premier, à 0 le bit de poids fort est émis en premier.

**MSTR Master/Slave Select** Mode de fonctionnement de l'interface : Si à 1, mode Maître, à 0 mode Escalve.

**CPOL Clock Polarity** Sélection de la polarité de l'horloge : Si à 1, repos de l'horloge au niveau haut (1), à 0 repos de l'horloge au niveau bas (0).

**CPHA Clock Phase** Sélection de la phase des données par rapport à l'horloge : si à 1, les données changent de front quand le signal d'horloge est au niveau bas, à 0 les données changent de front quand le signal d'horloge est au niveau haut.

**SPR1 & 2 SPI Clock Rate Select 1 & 2** Sélection de la fréquence de transmission, définit par le Maître :

SPI2X	SPR1	SPR2	Facteur de pré division
0	0	0	Fosc / 4
0	0	1	Fosc / 16
0	1	0	Fosc / 64
0	1	1	Fosc / 128
1	0	0	Fosc / 2
1	0	1	Fosc / 4
1	1	0	Fosc / 16
1	1	1	Fosc / 64

Le bit **SPI2X** est défini dans le registre **SPSR** qui suit.

### Registre SPSR (SPI Status Register)

Le registre de Statut du SPI.

Adresse	7	6	5	4	3	2	1	0
\$0E	<b>SPIF</b>	<b>WCOL</b>	-	-	-	-	-	<b>SPI2X</b>
L/E	L/E	L/E	L	L	L	L	L	L/E

**SPIF SPI Interrupt Flag** Drapeau d'indication de fin de transmission de l'octet. Mis à 1 en fin de transmission et mis à 0 après l'exécution de la routine d'interruption.

**WCOL Write Collision Flag** Bit d'Indication de collision de données, provoqué par l'écriture dans le tampon **SPDR** alors que le transfert en cours n'est pas terminé. La remise à 0 de ce bit est effectuée en lisant en premier lieu le registre **SPSR** quand ce bit est à 1, puis le registre **SPDR**.

**SPI2X SPI double Speed** Doublement de la vitesse en mode Maître. (Voir le registre **SPCR**).

### Registre SPDR (SPI Data Register)

Le registre de données.

Adresse	7	6	5	4	3	2	1	0
\$0F	<b>SPD7</b>	<b>SPD6</b>	<b>SPD5</b>	<b>SPD4</b>	<b>SPD3</b>	<b>SPD2</b>	<b>SPD1</b>	<b>SPD0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**SPD7:0 SPI Data Register** En écrivant dans ce registre, le début de la transmission est lancé. Il contient aussi le résultat de la dernière réception.

## Programmation de la SPI

Deux exemples sont présentés, le premier est une transmission en mode Maître et la seconde une réception en mode Esclave.

### Exemple de transmission SPI

```

SPI_MasterInit:                                ; Mes MOSI et SCK en sortie, le reste en entrée
    Ldi    r17,(1<<DD_MOSI)|(1<<DD_SCK)
    Out    DDR_SPI,r17
    ldi    r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
    out    SPCR,r17                            ; Active SPI, mode Maître, horloge à Fck/16
    ret

SPI_MasterTransmit:                            ; Début de transmission d'une donnée (r16)
    Out    SPDR,r16

Wait_Transmit:                                ; Attendre pour la transmission complète
    Sbis   SPSR,SPIF
    Rjmp  Wait_Transmit
    Ret
    
```

## Exemple de réception SPI

```
SPI_SlaveInit:                                ; Mettre MISO en sortie et le reste en entrée
    Ldi    r17,(1<<DD_MISO)
    out    DDR_SPI,r17
    ldi    r17,(1<<SPE)                        ; Active SPI
    out    SPCR,r17
    ret

SPI_SlaveReceive:                              ; Attendre pour la réception complète
    sbis   SPSR,SPIF
    rjmp   SPI_SlaveReceive
    in    r16,SPDR                             ; Lire la donnée reçue
    ret
```

# L'interface Série USART

L'USART est l'abréviation de *Universal Synchronous and Asynchronous Receiver and Transmitter*, soit Interface Série Synchrone et Asynchrone avec les caractéristiques suivantes :

- Générateur interne de fréquence de cadencement
- Asynchrone et Synchrone opération
- Horloge Maître ou Esclave
- Echanges de données sur 5 à 9 bits et 1 ou 2 bits de stop
- Gestion des parités
- Communication en Full duplex (peut émettre et recevoir en même temps)
- Protection des débordements
- Détection de faux départs de transmission
- Filtrage de l'entrée
- Plusieurs interruptions programmables sur le mode émission et réception
- Mode double vitesse de communication

L'application principale de ce périphérique est la communication entre le microcontrôleur et un ordinateur via le port série **RS232**.

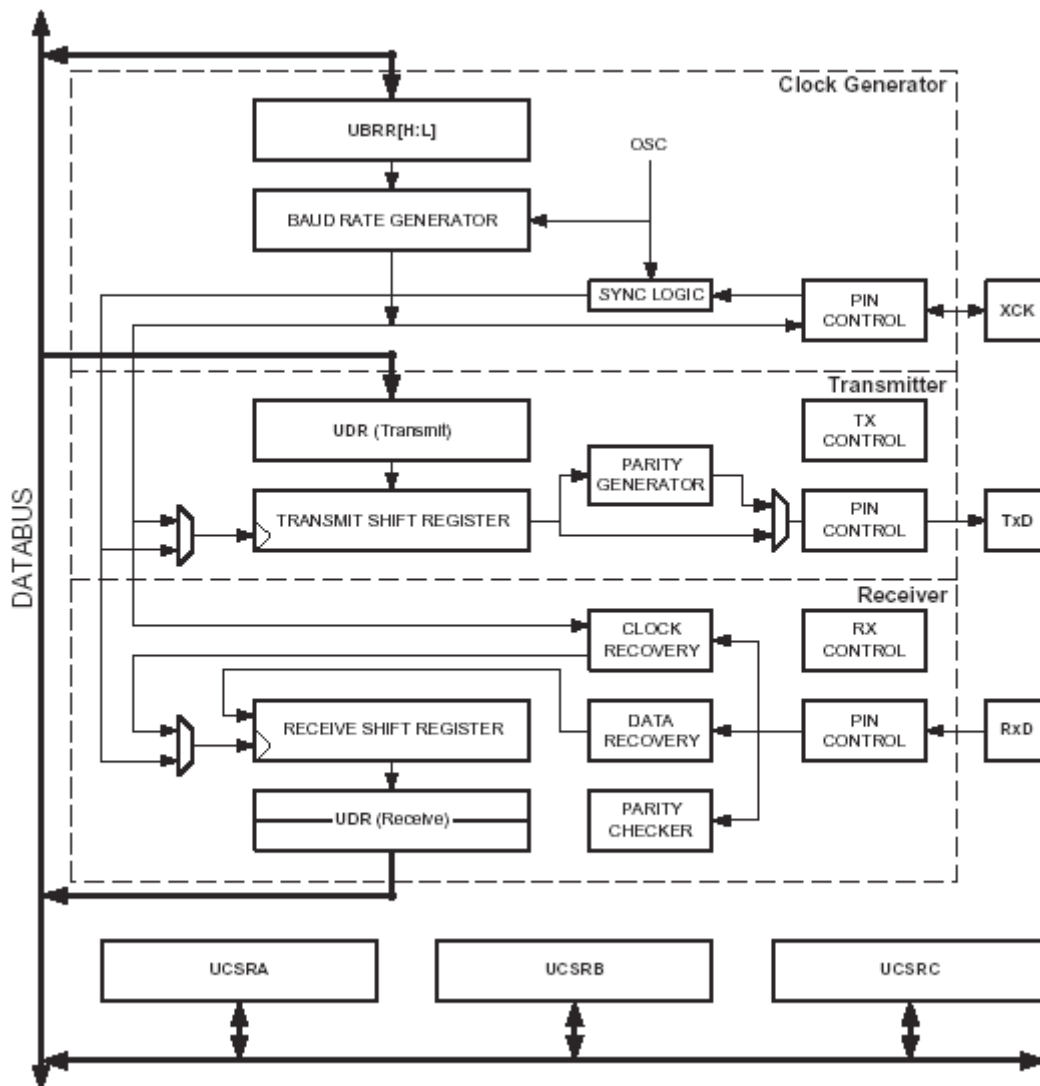


Figure 39, Synoptique de l'USART.

Les parties entourées dans le diagramme séparent les trois parties principales de l'USART : Générateur d'Horloge, Émetteur et Récepteur. Les registres de contrôle sont partagés par toutes les unités. La génération d'horloge consiste en une synchronisation pour l'entrée d'horloge externe employée par



l'opérateur Esclave Synchrone et le générateur de vitesse de transmission (en bauds). Le **XCK** (l'Horloge de Transfert) la broche est seulement employée par le mode de Transfert Synchrone. L'Émetteur consiste en un simple buffer écrit, un registre à décalage, le générateur de parité qui contrôle le traitement des formats d'encadrement. L'écriture du buffer permet un transfert en continu des données sans retard. Le Récepteur est la partie la plus complexe du module **USART** en raison de son horloge et des unités de rétablissement de données. Les unités de rétablissement sont employées pour la réception de données asynchrone. En plus des unités de rétablissement, le récepteur inclut un contrôleur paritaire, la logique de contrôle, un registre à décalage et deux niveaux de buffer de réception **UDR**. Le récepteur soutient les mêmes formats d'encadrement que l'émetteur et peut détecter l'erreur d'encadrement des données, des erreurs de débordement et paritaires.

## Génération d'Horloge

La logique de génération d'horloge produit l'horloge de base pour l'Émetteur et le Récepteur. **LUSART** soutient quatre modes d'opération d'horloge : la Vitesse Normale Asynchrone, Double Asynchrone, le Maître Synchrone et Esclave Synchrone. Le bit **UMSEL** dans le contrôle **USART** et le registre de statut **C** de **UCSRC** choisit entre l'opération asynchrone et synchrone. La vitesse double (mode Asynchrone seulement) est contrôlée par l'**U2X** dans le registre **UCSRA**. En employant le mode Synchrone (**UMSEL=1**), le registre de direction de données **DDR\_XCK** pour la broche **XCK** contrôle si la source d'horloge est interne (mode de Maître) ou externe (mode d'Esclave). La broche **XCK** est seulement active en employant le mode Synchrone.

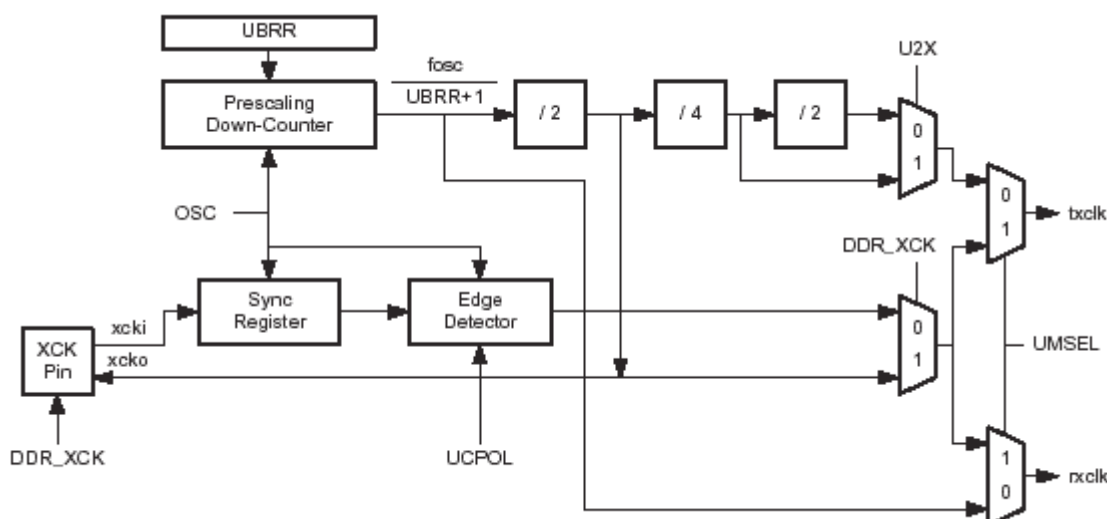


Figure 40, Synoptique de la gestion d'horloge de l'USART.

Description des signaux :

**txclk** horloge d'Émetteur (Signal Interne)

**rxclk** base de l'horloge du Récepteur (Signal Interne)

**xcki** Entrée de la broche **XCK** (Signal Interne). Employé pour Esclave Synchrone

**xcko** Sortie d'horloge sur la broche **XCK** (Signal Interne). Employé pour Maître Synchrone

**fosc XTAL** Fréquence de l'horloge de système.

## Génération d'Horloge Interne - Le Générateur de Vitesse de transmission (en bauds)

La génération d'horloge interne est employée pour les modes de fonctionnement asynchrone et Maître synchrone. Le registre de vitesse de transmission (bauds) **UBRR** et le compteur connecté à la même fonction qu'un pré diviseur programmable ou générateur de vitesse de transmission. Le compteur utilise l'horloge système **Fosc** qui est chargée dans **UBRR** chaque fois que compteur est arrivé à zéro ou quand le registre **UBRR** est écrit. Un top d'horloge est produit chaque fois que le compteur atteint 0. Ce top d'horloge correspond au générateur de vitesse de transmission (bauds) soit ( $Fosc / (UBRR+1)$ ). L'Émetteur divise le générateur de vitesse de transmission (bauds) par 2, 8 ou 16 selon le mode. Le générateur de vitesse de transmission est employé directement par le récepteur d'horloge et l'unité de rétablissement des

données. Cependant, les unités de rétablissement emploient une machine d'état et emploient 2, 8 ou 16 états selon la configuration des bits **UMSEL**, **U2X** et **DDR\_XCK**.

Operating Mode	Equation for Calculating Baud Rate <sup>(1)</sup>	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Figure 41, Calcul de la vitesse de transmission (Bauds).

### Opération de Vitesse Double (U2X)

Le taux de transfert peut être doublé en mettant le bit **U2X** à 1 dans **UCSRA** uniquement pour la communication asynchrone. Ce bit est à zéro en mode synchrone.

Ce bit réduira le diviseur d'horloge de 16 à 8, doublant efficacement le taux de transfert pour la communication asynchrone. Le récepteur utilisera dans ce cas la moitié du nombre d'échantillons (réduit de 16 à 8) pour l'échantillon de donnée et le rétablissement d'horloge et donc une plus grande précision sera demandée sur l'horloge système quand ce mode sera employé. Pour l'Émetteur, il n'y a aucun impacte.

### Horloge Externe

Le chronométrage externe est employé par les modes de fonctionnement Esclave Synchrones. L'entrée d'horloge externe de la broche **XCK** est échantillonnée par un registre de synchronisation pour réduire au minimum l'instabilité. La production du registre de synchronisation doit alors passer par un détecteur de front avant qu'il ne puisse être employé par l'Émetteur et le Récepteur. L'horloge **XCK** est limitée en fréquence par l'équation suivante :

$$f_{XCK} < \frac{f_{OSC}}{4}$$

### Opération d'Horloge Synchrone

Quand le mode Synchrone est employé **UMSEL**=1, la broche **XCK** sera employée ou bien comme entrée d'horloge (Esclave) ou bien pour la production d'horloge (Maître). La dépendance entre les fronts d'horloge et le prélèvement d'échantillons de données ou le changement de données est la même. Le principe de base est que l'entrée de données (sur **RxD**) est échantillonnée à l'opposé du front d'horloge **XCK** et du front de la donnée (**TxD**).

### Formats du mot de donnée

Un mot de donnée périodique est défini pour être un bit de données avec des bits de synchronisation (le début et des bits d'arrêt) et facultativement un bit de parité pour la vérification d'erreur. **LUSART** accepte 30 combinaisons possibles de mot de donnée avec :

- 1 bit de début
- 5, 6, 7, 8, ou 9 bits de données
- Parité : sans, impaire, paire
- 1 ou 2 bits d'arrêt

Un mot commence par un bit de début suivi par les bits de données du moins fort au plus fort, jusqu'à un total de neuf. S'il est demandé, le bit de parité est inséré après les bits de donnée, puis vient le bit ou les bits d'arrêt.

Quand un mot complet est transmis, il peut être directement suivi par un nouveau mot, ou la ligne de communication peut être mis à l'état haut (libre). La figure suivante illustre les combinaisons possibles des formats de mot de donnée. Les bits entre parenthèses sont facultatifs.

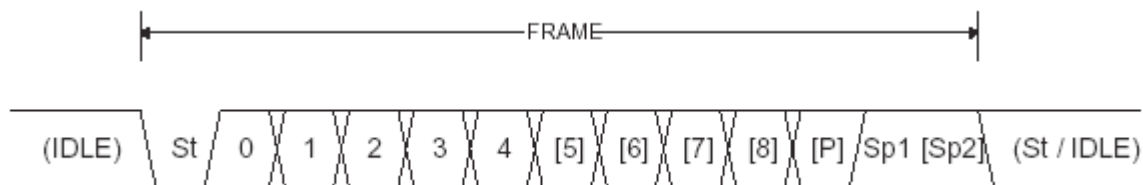


Figure 42, Trame d'un mot de donnée.

### Calcul du bit de Parité

Le bit de parité est calculé en faisant un 'ou exclusif' de tous les bits de donnés. Si la parité impaire est employée, le résultat du ou exclusif est inversé. La relation entre le bit de parité et des bits de donnés est la suivante :

$$P_{even} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{odd} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

**Peven** bit Paritaire employant même parité

**Podd** bit Paritaire employant parité étrange (impaire)

**dn** bit de Données n du caractère

Si le contrôle de parité est employé, le bit de parité est placé entre le dernier bit de donnés et avant le bit de stop dans le mot de donnée.

### Initialisation de l'USART

L'USART doit être initialisé avant qu'une communication ne puisse avoir lieu. Le processus d'initialisation consiste normalement à définir la vitesse de transmission en bauds, définir le format d'encadrement et choisir l'Émetteur ou le Récepteur selon l'utilisation. Pour les interruptions de l'USART, l'interruption globale doit être interrompue et généralement mis hors de service en faisant l'initialisation.

Avant de faire une réinitialisation avec la vitesse de transmission (en bauds) ou le format d'encadrement, soyez sur qu'il n'y a aucune transmission en cours. Le drapeau **TXC** peut être employé pour vérifier que l'Émetteur a achevé tous les transferts et le drapeau **RXC** peut être employé pour vérifier qu'il n'y a aucune donnée non lue dans le buffer du récepteur. Notez que le drapeau **TXC** doit être remis à 0 avant chaque transmission, avant qu'**UDR** ne soit écrit, s'il est employé à cette fin.

### Les Registres de l'USART

Les registres de l'USART sont aux nombres de cinq avec une particularité, car deux registres occupent la même adresse physique, **UBRRH** partage le même emplacement d'entrée-sortie que le Registre **UCSRC**.

#### Registre UCSRA (USART Control and Status Register A)

Le registre de contrôle et de statut de l'interface USART.

Adresse	7	6	5	4	3	2	1	0
\$0B	<b>RXC</b>	<b>TXC</b>	<b>UDRE</b>	<b>FE</b>	<b>DOR</b>	<b>PE</b>	<b>U2X</b>	<b>MPCM</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**RXC USART Receive Complete** Passe à 1 lorsque le tampon de réception contient une donnée. Ce bit est automatiquement remis à 0 lors de la lecture du tampon correspondant, soit **UDR**.

**TXC USART Transmit Complete** Passe à 1 lorsque la donnée contenue dans **UDR** à été transmise. Ce bit est automatiquement remis à 0 lors du traitement de l'interruption correspondante, ou alors il faut écrire un 1 par programme.

**UDRE USART Data Register Empty** Passe à un lorsque la donnée écrite dans **UDR** est passée en phase d'émission. ce drapeau signale donc le déplacement du contenu de **UDR** vers le tampon

d'émission. Repasse à 0 lors d'une nouvelle écriture de données dans **UDR**. **UDRE** est à 1 après un reset.

**FE USART Frame Error** Ce bit est mis à 1 sur lors d'un défaut de réception, en particulier quand le bit de stop n'a pas été réalisé correctement. La remise à 0 de ce bit se fait automatiquement lors de la prochaine réception correcte après le bit stop. Ce bit est mis à 0 par l'écriture dans **UCSRA**.

**DOR USART Data OverRun** Ce bit est mis à 1 lorsqu'une donnée précédemment reçue est disponible dans le registre **UDR**, cependant une autre donnée arrive dans le tampon de réception alors que **UDR** n'a toujours pas été lu. On peut donc dire qu'il s'agit d'un drapeau d'avertissement avant la future perte de données si rien n'est fait. La remise à 0 de ce bit est effectué automatiquement lors de la lecture du registre **UDR**. Il peut aussi s'agir d'une erreur de réception car le buffer est plein. Ce bit est mis à 0 par l'écriture dans **UCSRA**.

**PE USART Parity Error** Indique une erreur de parité lors de la réception d'un octet de donnée. Ce bit est valide après la réception du mot entier, soit **RXC** = 1. Ce bit est mis à 0 par l'écriture dans **UCSRA**.

**U2X Double the USART Transmission Speed** Ce bit n'est utilisable qu'avec l'interface Asynchrone, il est mis à 1 pour doubler la vitesse de transmission. En mode synchrone il est lu à 0.

**MPCM Multi Processor Communication Mode** Mode de communication multiprocesseur si à 1.

### Registre UCSRB (USART Control and Status Register B)

Le registre de contrôle et d'état de l'interface **USART** deuxième partie.

Adresse	7	6	5	4	3	2	1	0
\$0A	<b>RXCIE</b>	<b>TXCIE</b>	<b>UDRIE</b>	<b>RXEN</b>	<b>TXEN</b>	<b>UCSZ2</b>	<b>RXB8</b>	<b>TXB8</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**RXCIE RX Complete Interrupt Enable** Écrit le bit à 1 permet de prendre en compte les interruptions sur la réception complète dans **RXC**.

**TXCIE TX Complete Interrupt Enable** Écrit le bit à 1 permet de prendre en compte les interruptions sur la transmission complète dans **TXC**.

**UDRIE USART Data Register Empty Interrupt Enable** Écrit ce bit à 1 permet de prendre en compte les interruptions lorsque le registre d'émission est en cours de transfert dans **UBR**.

**RXEN Receiver Enable** Activation de réception de l'**USART** quand mis à 1. La mise hors service du Récepteur mettra à 0 les bits **FE**, **DOR** et **PE**.

**TXEN Transmitter Enable** Activation de la transmission de l'**USART** quand mis à 1. La mise hors service de l'Émetteur, mise à 0, deviendra efficace après la fin des transmissions en cours et en suspens, c'est-à-dire, quand le registre de chargement et le registre du buffer ne contiendra plus de données à transmettre.

**UCSZ2 Character Size** Le bit **UCSZ2** combiné avec le bit **UCSZ1:0** dans **UCSRC** indique le nombre de bit de donnée à émettre et à recevoir.

**RXB8 Receive Data bit 8** C'est le neuvième bit de donnée du caractère reçu en faisant fonctionner avec neuf bits de donnée. Il doit être lu avant la lecture des bits bas dans l'**UDR**.

**TXB8 Transmit Data Bit 8** C'est le neuvième bit de donnée dans le caractère à transmettre en faisant fonctionner avec neuf bits de donnée. Il doit être écrit avant l'écriture des bits bas dans **UDR**.

### Registre UCSRC (USART Control and Status Register C)

Le registre de contrôle et d'état de l'interface **USART** troisième partie. Le Registre **UCSRC** partage le même emplacement d'entrée-sortie que le Registre **UBRRH**.

Adresse	7	6	5	4	3	2	1	0
\$20	<b>URSEL</b>	<b>UMSEL</b>	<b>UPM1</b>	<b>UPM0</b>	<b>USBS</b>	<b>UCSZ1</b>	<b>UCSZ0</b>	<b>UCPOL</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**URSEL USART Register Selecter** Sélection du registre entre **UBRRH** et **UCSRC**. Il est lu à 1 quand on lie **UCSRC** et il faut le mettre à 1 pour écrire dans **UCSRC**.

**UMSEL USART Mode Select** Ce bit sélectionne entre le mode Asynchrone à 0 et Synchrone à 1.

**UPM1:0 Parity Mode** Ces bits sélectionnent le type de parité et le contrôle associé. Si c'est permis, l'émetteur la produira automatiquement et l'enverra après les bits donnés. Le Récepteur produira une valeur de parité pour les données entrantes et la comparera avec **UPM1:0**. Si une erreur est détectée, le drapeau **PE** dans **UCSRA** sera mis à 1.

UPM1	UPM0	Mode de Parité
0	0	Désactivé
0	1	Réservé
1	0	Parité Paire (Even)
1	1	Parité Impaire (Odd)

**USBS Stop Bit Select** Ce bit sélectionne le nombre de bit de Stop qu'il faut insérer à la fin de la transmission. Si à 0 1 bit de Spot, si à 1 deux bit de Stop.

**UCSZ1:0 Character Size** Les bits **UCSZ1:0** combinées avec le bit **UCSZ2** dans **UCSRB** détermine le nombre de bit de la donnée dans un mot à émettre et à recevoir :

UCSZ2	UCSZ1	UCSZ0	Taille du caractère
0	0	0	5 bits
0	0	1	6 bits
0	1	0	7 bits
0	1	1	8 bits
1	0	0	Réservé
1	0	1	Réservé
1	1	0	Réservé
1	1	1	9 bits

**UCPOL Clock Polarity** Ce bit est employé pour le mode Synchrone seulement. Écrivez ce bit à 0 quand le mode Asynchrone est employé. Le bit **UCPOL** met le rapport entre le changement de production de donnée, l'échantillon d'entrée de données et l'horloge synchrone **XCK**.

UCPOL	Donnée Transmise (Sortie sur la broche TxD)	Donnée Reçue Échantillonnée (Entrée sur la broche RxD)
0	Sur Front Montant de <b>XCK</b>	Sur Front Descendant de <b>XCK</b>
1	Sur Front Descendant de <b>XCK</b>	Sur Front Montant <b>XCK</b>

### Registre UDR (USART I/O Data Register)

Le registre de données de l'interface USART.

Adresse	7	6	5	4	3	2	1	0
\$0C	UDR7	UDR6	UDR5	UDR4	UDR3	UDR2	UDR1	UDR0
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**UDR7:0 USART I/O Data Register** En fait ce registre est assez particulier, la gestion est transparente pour l'utilisateur, mais sachez que lors d'une opération d'écriture de données qui lance le processus d'émission, on peut lire aussitôt ce registre qui contiendra le tampon de réception.

### Registre UBRR (USART Baud Rate Register)

Le registre de sélection de la vitesse de cadencement de l'USART.

Le Registre **UBRRH** partage le même emplacement d'entrée-sortie que le Registre **UCSRC**. Voir le registre **UCSRC**.

Adresse	15	14	13	12	11	10	9	8
\$20	URSEL	-	-	-	UBRR11	UBRR10	UBRR9	UBRR8
Adresse	7	6	5	4	3	2	1	0
\$09	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**URSEL USART Register Selecter** Sélection du registre entre **UBRRH** et **UCSRC**. Il est lu à 0 quand on lie **UBRRH** et il faut le mettre à 0 pour écrire dans **UBRRH**.

**UBRR11:0 USART Baud Rate Register** Valeur permettant de sélectionner la fréquence de cadencement de l'USART. Il est à signaler avant tout que la fréquence de cadencement ne doit pas forcément correspondre exactement aux normes en vigueur, tout système à une tolérance concernant les fréquences d'émissions et réceptions. D'ailleurs le microcontrôleurs lui-même effectue un triple échantillonnage du signal a sur un demi-flanc, ce qui confirme que le système est insensible aux fréquences qui dérivent autour de la norme, et permet de limiter les erreurs dues aux parasites.

Voici quatre figures contenant les valeurs de **UBRR** en fonctions de la valeur du Quartz. Les valeurs **UBRR** sont bien sur données en décimales.

Baud Rate (bps)	$f_{osc} = 1.0000 \text{ MHz}$				$f_{osc} = 1.8432 \text{ MHz}$				$f_{osc} = 2.0000 \text{ MHz}$			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	-	-	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	-	-	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	-	-	-	-	-	-	0	0.0%	-	-	-	-
250k	-	-	-	-	-	-	-	-	-	-	0	0.0%
Max <sup>(1)</sup>	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

Tableau 1 des vitesses en baud par rapport à la Fréquence.

Baud Rate (bps)	$f_{osc} = 3.6864 \text{ MHz}$				$f_{osc} = 4.0000 \text{ MHz}$				$f_{osc} = 7.3728 \text{ MHz}$			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	-	-	0	-7.8%	-	-	0	0.0%	0	-7.8%	1	-7.8%
1M	-	-	-	-	-	-	-	-	-	-	0	-7.8%
Max <sup>(1)</sup>	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

Tableau 2 des vitesses en baud par rapport à la Fréquence.

Baud Rate (bps)	$f_{osc} = 8.0000 \text{ MHz}$				$f_{osc} = 11.0592 \text{ MHz}$				$f_{osc} = 14.7456 \text{ MHz}$			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	-	-	2	-7.8%	1	-7.8%	3	-7.8%
1M	-	-	0	0.0%	-	-	-	-	0	-7.8%	1	-7.8%
Max <sup>(1)</sup>	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

Tableau 3 des vitesses en baud par rapport à la Fréquence.

Baud Rate (bps)	$f_{osc} = 16.0000 \text{ MHz}$				$f_{osc} = 18.4320 \text{ MHz}$				$f_{osc} = 20.0000 \text{ MHz}$			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	-	-	4	-7.8%	-	-	4	0.0%
1M	0	0.0%	1	0.0%	-	-	-	-	-	-	-	-
Max <sup>(1)</sup>	1 Mbps		2 Mbps		1.152 Mbps		2.304 Mbps		1.25 Mbps		2.5 Mbps	

Tableau 4 des vitesses en baud par rapport à la Fréquence.

## Programmation de l'USART

### Paramétrage de l'USART

Voici les principaux modes de fonctionnement :

Initialisation    Emission    Réception

Mise en marche et validation de l'USART avec **UCR**

Configuration de la vitesse de l'USART avec **UBRR RAZ** de **TXC** dans **USR**

Emission de la donnée contenue dans **UDR**

Attente de **TXC** = 1 pour fin d'émission  
Lecture de la donnée contenue dans **UDR**

Attente de **RXC** = 1 pour fin de réception

### Exemple de programmation

Les exemples de code d'initialisation suivants montrent une opération asynchrone employant un format d'encadrement fixé. Le paramètre de vitesse de transmission (en bauds) est assumé pour être stocké dans les registres r17:r16.

```
USART_Init:                                ; Choix de la vitesse en baud
out    UBRRH,r17
out    UBRRL,r16
ldi    r16,(1<<RXEN)|(1<<TXEN)
out    UCSRB,r16        ; Active Réception et Transmission
ldi    r16,(1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
out    UCSRC,r16        ; Trame 8 bits, pas parité, 2 stop
ret
```

Transmission de 5 à 8 bit.

```
USART_Transmit:                             ; Attendre le buffer vide pour transmettre
sbis   UCSRA,UDRE
rjmp   USART_Transmit
out    UDR,r16        ; Envoie la donnée dans le buffer et transmettre
ret
```

Si des caractères à 9 bits sont employés **UCSZ=7**, la neuvième bit doit être écrite sur le bit **TXB8**.

```
USART_Transmit:                             ; Attendre le buffer vide pour transmettre
sbis   UCSRA,UDRE
rjmp   USART_Transmit
cbi    UCSRB,TXB8    ; Copy 9ème bit sur TXB8
sbrc   r17,0
sbi    UCSRB,TXB8
out    UDR,r16        ; Met LSB donnée (r16) dans le buffer et envoie
ret
```

L'exemple de code suivant montre la réception basée sur le drapeau **RXC**. En employant des mots de moins de 8 bits, les bits les plus significatifs des données lues sur **UDR** seront marqués à 0. L'**USART** doit être initialisé avant que la fonction ne soit employée.

Réception de 5 à 8 bits.

```
USART_Receive:                              ; Attendre la réception d'une donnée
Sbis   UCSRA,RXC
Rjmp   USART_Receive
in     r16,UDR        ; Retour avec la donnée dans r16.
ret
```

Si des caractères à 9 bits sont employés **UCSZ=7**, le neuvième bit doit être lue et mis dans le bit **TXB8**.

```
USART_Receive:                              ; Attendre la réception d'une donnée
Sbis   UCSRA,RXC
Rjmp   USART_Receive
in     r18,UCSRA      ; Regarde le statut et le 9ème bit du buffer
in     r17,UCSRB
in     r16,UDR
andi   r18,(1<<FE)|(1<<DOR)|(1<<PE)
breq   USART_ReceiveNoError ; Si erreur -> retourne avec -1
ldi    r17,HIGH(-1)
ldi    r16,LOW(-1)

USART_ReceiveNoError:                       ; Filtre le 9ème bit
Lsr    r17
Andi   r17,0x01
ret
```



## L'interface I2C (TWI)

L'interface à deux conducteurs périodique **TWI** est un dérivé de l'interface **I2C** de Philips, c'est une nouveauté du modèle **ATMEGA**, il n'existe pas dans les versions précédentes **AT89** et **AT90**.

Le protocole **TWI** permet de connecter jusqu'à 111 systèmes différents employant seulement deux lignes de bus bidirectionnelles, un pour l'horloge **SCL** et un pour les données **SDA**. Le seul matériel externe nécessaire pour la mise en oeuvre du bus est un simple jeu de résistance pour chacune des lignes (**R1**, **R2**). Tous les systèmes connectés au bus ont des adresses individuelles et les mécanismes de control sont inhérents au protocole **I2C**.

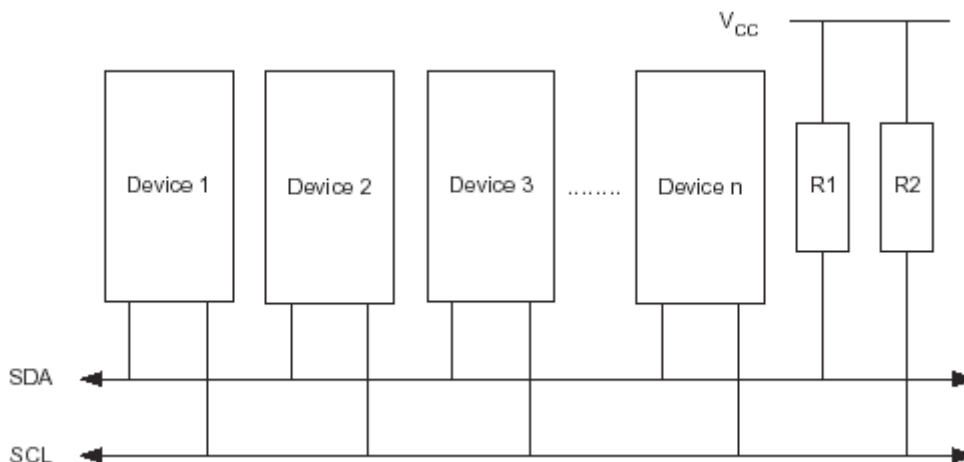


Figure 43, Principe du bus **I2C** ou **TWI**.

Comme pour une interface **SPI**, des termes spécifiques sont utilisés :

Terme	Description
Maître	Le système qui amorce et termine une transmission. Le Maître produit aussi l'horloge <b>SCL</b> .
Esclave	Le système adressé par un Maître.
Émetteur	Le système plaçant la donnée sur le bus.
Récepteur	Le système lisant la donnée sur le bus.
<b>SCL</b>	Horloge de synchronisation du bus
<b>SDA</b>	Donnée transmise ou reçus du bus

### Intercommunication Électrique

Comme le montre la figure ci-dessus, les deux lignes du bus sont connectées à la tension positive par des résistances de tirage d'une valeur de 4,7 à 5 **Kohm**. Tous les systèmes **I2C** Esclave sont en drain ouvert et exécutent une fonction 'ET' câblé qui est essentiel à l'interface. Un niveau bas sur une ligne du bus **TWI** est produit quand un ou plusieurs systèmes **TWI** sont commandés. Un haut niveau est produit quand tous les systèmes **TWI** sont en mode tri états, permettant aux résistances de tirer la ligne vers le haut.

Le nombre de système qui peut être connecté au bus est limité par la capacité du bus qui doit être inférieur à 400 pF et par l'espace d'adresse des Esclave qui est limité à 7 bits.

La fréquence du bus est soit inférieur à 100 **KHz**, soit de 100 **KHz** à 400 **KHz**.

### Transfert de Données et Format d'Encadrement

#### Transfert de Bit

Chaque bit de donnée transférée sur le bus **TWI** est accompagné par une impulsion sur la ligne d'horloge **SCL**.

Le niveau de la ligne de données doit être stable quand la ligne d'horloge est haute. La seule exception de cette règle est la production des conditions d'arrêt et le début (**START** et **STOP**).

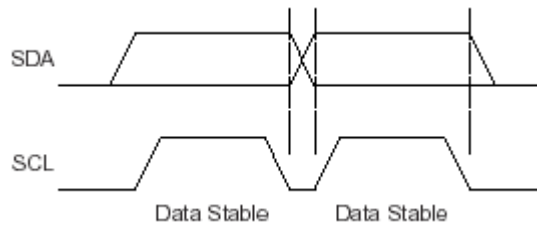


Figure 44, Stabilité des échanges du bus.

## Condition de DÉBUT et d'ARRÊT (START & STOP)

Le Maître amorce et termine une transmission de données. La transmission est amorcée quand le Maître envoie une condition de **DÉBUT** sur le bus et elle est terminée quand le Maître envoie une condition d'**ARRÊT**. Entre un **DÉBUT** et un **ARRÊT**, le bus est considéré occupé et aucun autre Maître ne doit saisir le contrôle du bus. Un cas spécial arrive quand une nouvelle condition de **DÉBUT** est envoyée entre un **DÉBUT** et un **ARRÊT**, c'est une condition de **DÉBUT RÉPÉTÉE** (ReStart) qui est employé quand le Maître veut amorcer un nouveau transfert sans perdre le contrôle du bus. Après un **DÉBUT RÉPÉTÉ**, le bus reste occupé jusqu'à l'**ARRÊT** suivant et le fonctionnement est identique à la condition de **DÉBUT**.

Comme le décrit la figure ci-dessous, la condition de **DÉBUT** et d'**ARRÊT** est signalé par le changement de niveau de la ligne **SDA** quand la ligne **SCL** est haute.

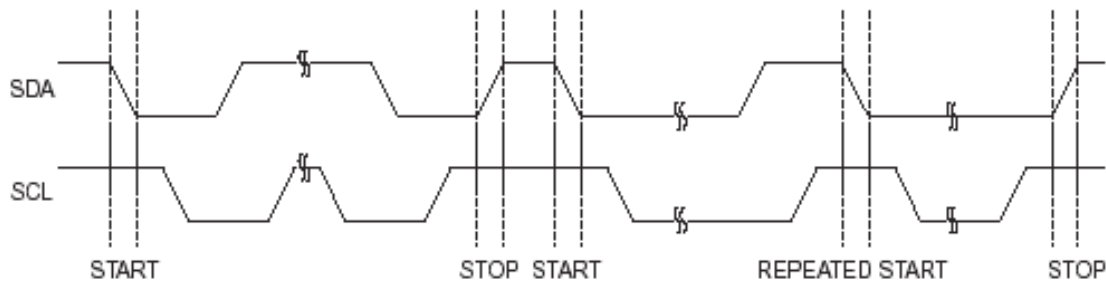


Figure 45, Conditions de **START**, **STOP** et **RESTART**.

La chute de la ligne **SDA** signale un **DÉBUT** ou un **DÉBUT RÉPÉTÉE** et la montée de la ligne **SDA** signal un **ARRÊT** quant le ligne **SCL** est stable à l'état haut.

La transmission des autres paquets d'informations ce fait par la modification de la ligne **SCL** quant la ligne **SDA** est stable (donc l'inverse).

## Format du Paquet d'Adresse

Tous les paquets d'adresse transmis sur le bus ont neuf bits de long. Le paquet d'adresse contient les sept bits d'adresse proprement dite, un bit de contrôle pour indiquer s'il s'agit d'une Lecture ou d'une Ecriture **R/W** et un bit de reconnaissance **ACK**.

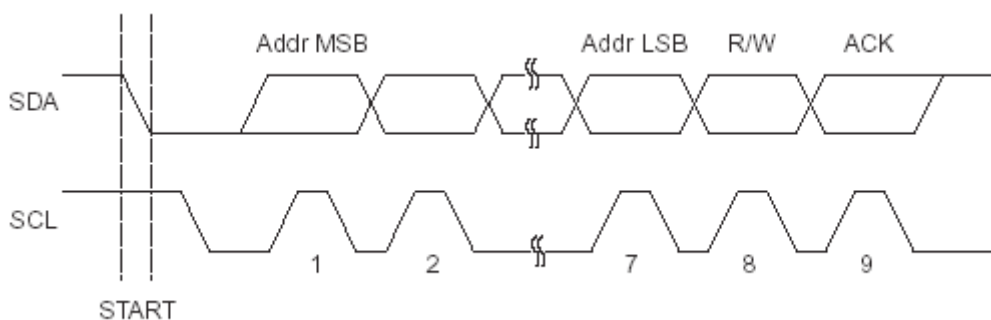


Figure 46, Format de l'Adresse.

L'octet d'adresse commence toujours par le poids fort **MSB** qui est transmis en premier suivi des 6 bits restant dans l'ordre, et finie donc par le poids faible **LSB**.

Si le bit **R/W** est mis à 1, une opération de lecture doit être exécutée, autrement à 0, une opération d'écriture doit être exécutée.

Quand un Esclave reconnaît qu'il est adressé, il doit se signaler en tirant **SDA** à l'état bas sur le neuvième cycle d'horloge **SCL**, le bit **ACK**. Si l'Esclave adressé est occupé, ou ne peut pas répondre à la demande du Maître, la ligne **SDA** restera à l'état haut sur le cycle d'horloge **ACK**. Le Maître peut alors transmettre une condition **d'ARRÊT**, ou une condition de **DÉBUT RÉPÉTÉE** pour amorcer une nouvelle transmission. Un paquet d'adresse constituer d'une adresse d'Esclave **LU** ou **ÉCRIT** est appelé **SLA+R** ou **SLA+W**, respectivement. L'adresse de l'Esclave peut librement être allouée par le programmeur, sauf l'adresse **b0000000** qui est réservée pour un appel général.

### Appel Général des Esclaves

Quand un appel général est envoyé (Adresse **b0000000**), tous les Esclaves doivent répondre en tirant la ligne **SDA** à l'état bas dans le cycle **ACK**. Un appel général est employé quand un Maître veut transmettre le même message à tous les Esclaves du système. Quand l'adresse d'appel générale suivie par un bit d'Écriture est transmise sur le bus, tous les Esclaves tirent la ligne **SDA** à l'état bas dans le cycle **ACK**. Les paquets de données suivants seront alors reçus par tous les Esclaves qui ont reconnu l'appel général.

**Note :** La transmission de l'adresse d'appel générale suivie par un bit Lu est sans signification.

**Note 2 :** Toutes les adresses aux formats **b1111xxx** doivent être réservées pour un usage futur.

### Format du Paquet de Données

Tous les paquets de données transmis sur le bus ont neuf bits de long, constituer d'un octet de données et une bit se reconnaissant.

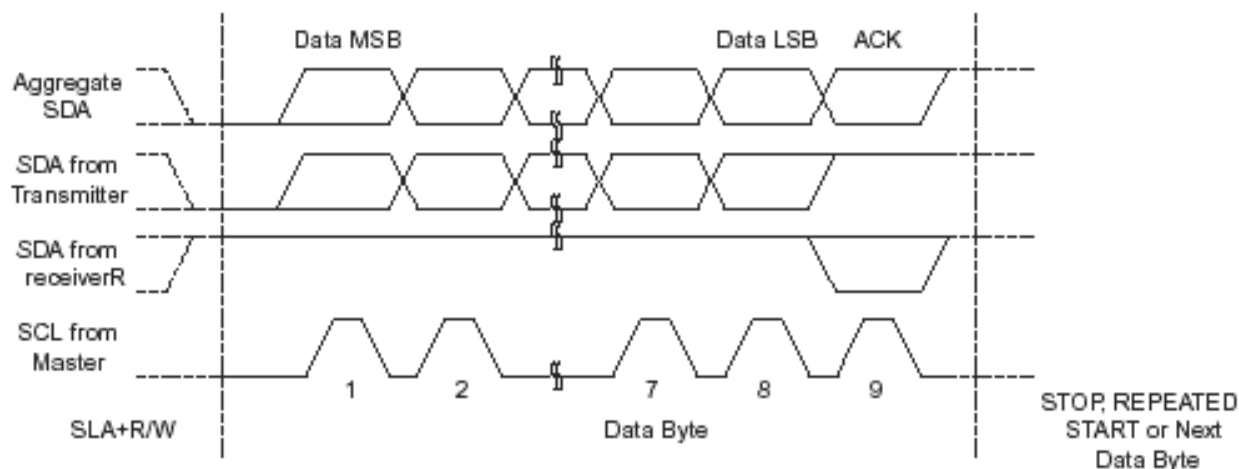


Figure 47, Format du paquet de Donnée.

Pendant un transfert de données, le Maître produit l'horloge et les conditions de **DÉBUT** et **d'ARRÊT**, tandis que le récepteur est responsable de reconnaître la réception par le signal **ACK**.

En reconnaissant le signal, le récepteur tire la ligne **SDA** à l'état bas sur le neuvième cycle **SCL**. Si le récepteur laisse la ligne **SDA** à l'état haut, un **non-ACK** ou **NACK** est signalé. Quand le récepteur a reçu le dernier octet d'une transmission et qu'il ne désire plus recevoir d'octet, il doit informer l'émetteur en envoyant un **NACK** après l'octet final. Le poids fort **MSB** de l'octet de données est transmis en premier.

### Transmission Type

Une transmission consiste essentiellement en une condition de **DÉBUT**, l'adresse d'un module Esclave **SLA+R/W**, un ou plusieurs paquets de données et une condition **d'ARRÊT**. Un message vide, constituer d'un **DÉBUT** suivi par un **ARRÊT**, est illégal.

Le 'ET' câblé de la ligne **SCL** peut être employé pour mettre en œuvre le système dit '*à poignée de mains*' entre le Maître et l'Esclave. L'Esclave peut prolonger la période du signal **SCL** à l'état bas en tirant la ligne **SCL** à l'état bas. Cela est utile si l'horloge émise par le Maître est trop rapide pour l'Esclave, ou l'Esclave a besoin du temps supplémentaire pour le traitement entre les transmissions de données. L'Esclave prolongeant la période du signal **SCL** à l'état bas n'affectera pas la période **SCL** à l'état haut, qui est

déterminée par le Maître. En conséquence, l'Esclave peut réduire la vitesse de transfert de données en prolongeant le cycle d'horloge **SCL**.

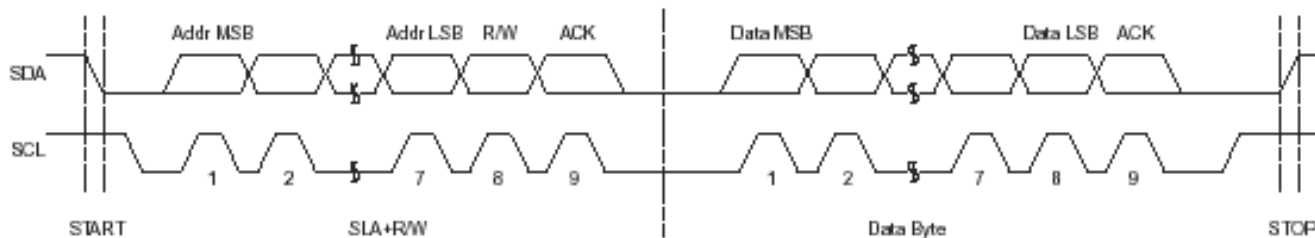


Figure 48, Transmission typique sur bus **TWI** ou **I2C**.

Plusieurs octets de données peuvent être transmis entre le **SLA+R/W** et la condition **d'ARRÊT**, selon le protocole de l'Esclave mis en oeuvre par votre logiciel d'application.

## Bus Multi-Maître, Arbitrage et Synchronisation

Le protocole du bus permet de fonctionner avec des systèmes contenant plusieurs Maîtres. Des conditions spéciales ont été développées pour assurer le fonctionnement normal, même si deux ou plusieurs Maîtres amorcent une transmission en même temps.

Deux problèmes surgissent dans les systèmes Multi Maître :

- Un algorithme doit être mis en oeuvre permettant seulement à un des Maîtres d'achever la transmission. Tous les autres Maîtres doivent cesser la transmission quand ils découvrent qu'ils ont perdu le processus de choix. Ce processus de choix est appelé l'arbitrage. Quand un Maître découvre qu'il a perdu le processus d'arbitrage, il doit immédiatement commuter pour travailler en mode Esclave pour vérifier s'il est adressé par le Maître gagnant. Le fait que des Maîtres multiples ont commencés la transmission en même temps ne doit pas être détectable par les Esclaves, c'est-à-dire que les données transférées sur le bus ne doivent pas être corrompues.
- Des Maîtres différents peuvent employer des fréquences **SCL** différentes. Un arrangement doit être inventé pour synchroniser les horloges de tous les Maîtres, pour laisser la transmission passer d'une façon inflexible. Cela facilitera le processus d'arbitrage.

Le 'ET' câblé des lignes du bus est employé pour résoudre ces deux problèmes. Les horloges de tous les Maîtres seront des 'ET' câblés, transférant une horloge combinée avec la période la plus haute qui sera égale à celui du Maître qui à la plus courte période. La période basse de l'horloge combinée est égale à la période basse du Maître avec la plus long période basse.

Tous les Maîtres doivent écouter la ligne **SCL**, avant de commencer à générer leur **SCL** propre pour éviter les conflits.

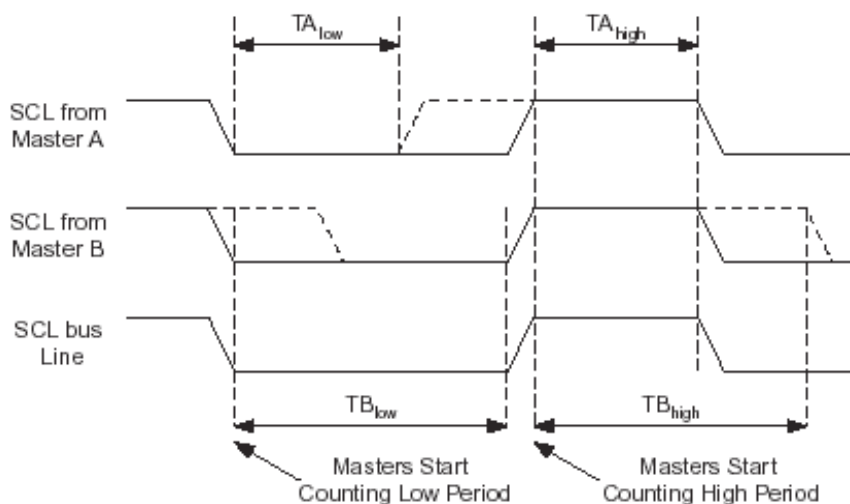


Figure 49, Synchronisation entre plusieurs Maître.

L'arbitrage est effectué par tous les Maîtres qui contrôlent continuellement la ligne **SDA** après la sortie des données. Si la valeur lue de la ligne **SDA** ne correspond pas à la valeur que le Maître devait attendre, il a

perdu l'arbitrage. Notez qu'un Maître peut perdre l'arbitrage seulement quand il produit un signal **SDA** à l'état haut tandis qu'un autre Maître produit un signal à l'état bas. Le Maître perdant doit immédiatement travailler comme un Esclave, et vérifier qu'il n'est pas adressé par le Maître gagnant. La ligne **SDA** doit être laissée haut, mais il est permis au Maître perdant de produire le signal d'horloge jusqu'à la fin des données actuelles ou du paquet d'adresse. L'arbitrage continuera jusqu'à ce qu'il ne reste plus de Maître et cela peut prendre beaucoup de bits. Si plusieurs Maîtres essayent d'adresser le même Esclave, l'arbitrage continuera dans le paquet de données.

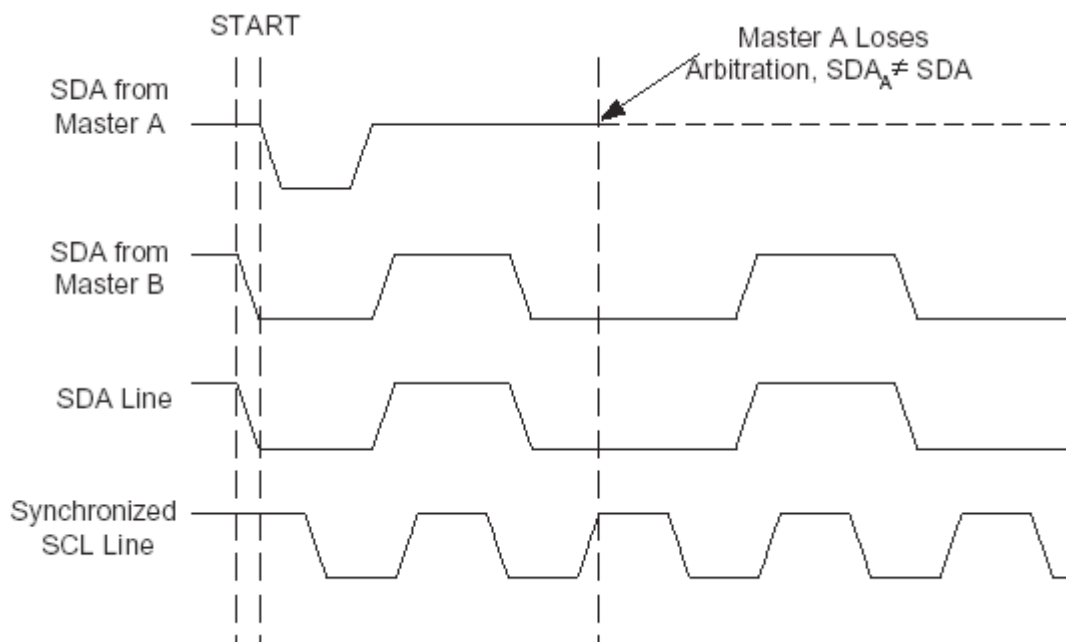


Figure 50, Arbitrage entre deux Maîtres.

L'arbitrage est interdit entre :

Une condition **DÉBUT RÉPÉTÉ** et un bit de données

Une condition **ARRÊT** et un bit de données

Une condition **DÉBUT RÉPÉTÉ** et une condition d'**ARRÊT**

C'est de la responsabilité du logiciel utilisateur d'assurer que les conditions d'arbitrage illégales n'arrivent pas. Cela implique dans des systèmes à multi Maître, que tous les transferts de données doivent employer la même composition de paquets de données et d'adresse. Autrement dit : Toutes les transmissions doivent contenir le même nombre de paquets de données, autrement le résultat de l'arbitrage est non défini.

## Vu d'ensemble du Module TWI (I2C)

Le module **TWI** comprend plusieurs sous modules, comme l'indiqué la figure suivante. Tous les registres dessinés en ligne épaisse sont accessibles par le bus de données **ATMEGA**.

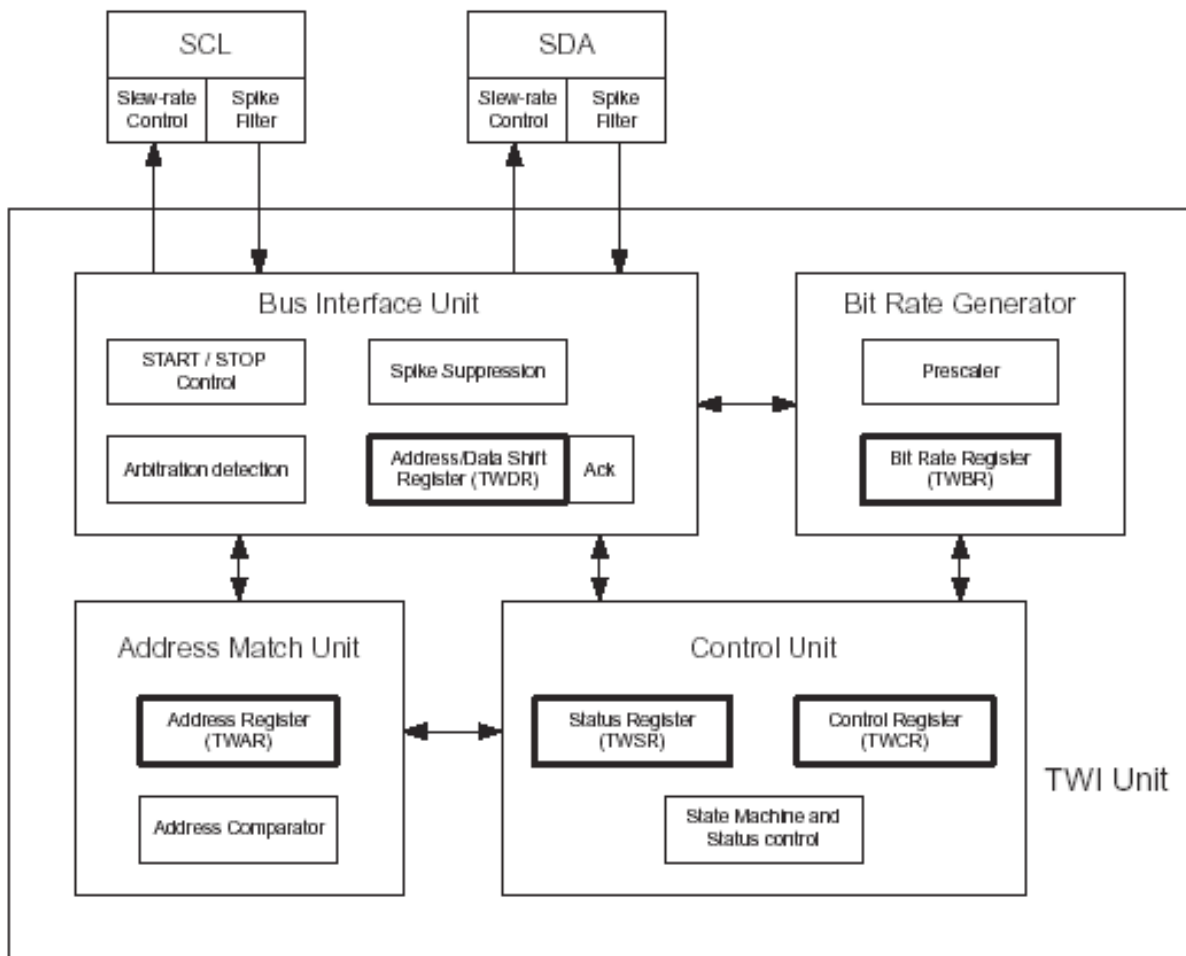


Figure 51, Synoptique du module **TWI**.

### Broche SCL et SDA

Ces broches interface le module **TWI** avec le reste du système **MCU**. L'interface **TWI** contient : un limiteur de vitesse pour se conformer à la spécification **I2C** (maxi de 400 KHz) et une unité de suppression de signaux de pointe enlevant des pointes inférieures à 50 ns. Les broches **SCL** et **SDA** correspondent respectivement aux ports **PC0** et **PC1**.

### Unité de Générateur de Taux de Bit

Cette unité contrôle la période d'horloge **SCL** en mode Maître. La période d'horloge **SCL** est contrôlée par le registre de taux de bit **TWBR** et les bits du pré diviseur dans le registre de statut **TWSR**.

Le mode Esclave ne dépend pas du taux de bit, ni du pré diviseur, mais de la fréquence d'horloge système qui doit être au moins 16 fois plus rapide que la fréquence **SCL** du Maître émetteur. Les Esclaves peuvent prolonger la période d'horloge **SCL** à l'état bas, réduisant ainsi la moyenne de l'horloge de bus. La fréquence **SCL** est produite selon l'équation suivante :

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{\text{TWPS}}}$$

Avec **TWBR** = valeur du registre de taux de bit et **TWPS** = Valeur des bits du pré diviseur dans le registre de statut.

**TWBR** doit être au minimum à **10** si l'interface **TWI** fonctionne en mode Maître. Si **TWBR** est inférieur à **10**, le Maître peut produire des signaux incorrects sur **SDA** et **SCL**.

### Unité d'Interface de bus

L'unité contient les données et le registre de changement d'adresse **TWDR**, un contrôleur de **DÉBUT/ARRÊT** et le matériel de détection d'arbitrage. Le **TWDR** contient l'adresse ou les octets de données à être transmis, ou l'adresse ou les octets de données reçus. En plus du registre **TWDR** à 8 bits, l'unité d'interface de bus contient aussi un registre contenant le bit **ACK** ou **NACK** à être transmis ou reçu. Le bit **ACK** ou **NACK** n'est pas directement accessible par le logiciel d'application. Cependant, une fois reçu, il peut être mis à 1 ou à 0 en manipulant le registre de contrôle **TWCR**. Quand dans le mode Émetteur est actif, la valeur du bit reçu **ACK** ou **NACK** peut être déterminée par la valeur dans **TWSR**.

Le contrôleur de **DÉBUT/ARRÊT** est responsable de la génération et la détection des conditions : **DÉBUT**, **DÉBUT RÉPÉTÉ** et **ARRÊT**, même quand le **MCU** est dans l'un des modes sommeil, permettant au **MCU** de se réveiller si il est adressé par un Maître.

Si le **TWI** a amorcé une transmission comme Maître, le matériel de détection d'arbitrage contrôle continuellement la transmission essayant de déterminer si l'arbitrage est dans le bon processus. Si le **TWI** a perdu l'arbitrage, l'unité de commande est informée et l'action correcte peut alors être prise et des codes de statut appropriés produits.

### Unité de Comparaison d'Adresse

L'unité de comparaison d'adresse vérifie si les octets d'adresse reçus correspondent à l'adresse à 7 bits du registre d'adresse **TWAR**. Si l'identification d'appel général permet le bit **TWGCE** dans **TWAR** est à 1, tous les octets d'adresse entrants seront aussi comparés avec l'adresse d'appel général. Sur une comparaison d'adresse, l'unité de commande est informée et l'action correcte sera pris. Le **TWI** peut ou ne peut pas reconnaître son adresse, selon **TWCR**. L'unité de comparaison d'adresse est capable de comparer des adresses même quand le **MCU** est en mode sommeil, permettant au **MCU** de se réveiller s'il est adressé par un Maître.

### Unité de Commande

L'unité de commande contrôle le bus **TWI** et produit les réponses correspondant aux conditions du registre de contrôle **TWCR**. Quand un événement exige l'attention de l'application sur le bus, le drapeau d'interruption **TWINT** est mis à 1. Dans le cycle d'horloge suivant, le registre de statut **TWSR** est mis à jour avec un code de statut identifiant l'événement. Le **TWSR** contient seulement l'information de statut appropriée quand le drapeau d'interruption est positionné. Le reste du temps **TWSR** contient un code de statut spécial indiquant qu'aucune information de statut appropriée n'est disponible. Tant que le drapeau **TWINT** est mis à 1, la ligne **SCL** est tenue à l'état bas. Cela permet au logiciel d'application d'achever ses tâches avant de reprendre la transmission **TWI** pour continuer.

Le drapeau **TWINT** peut survenir sur les situations suivantes, après que la l'interface **TWI** :

- A transmis la condition de **DÉBUT** ou **DÉBUT RÉPÉTÉ**,
- A transmis l'adresse **SLA+R/W**,
- A perdu l'arbitrage,
- A été adressé par sa propre adresse Esclave ou l'appel général,
- A reçu un octet de données,
- Après un **ARRÊT** ou un **DÉBUT RÉPÉTÉ** reçu en mode Esclave,
- Sur une erreur de bus arrivée en raison d'un **DÉBUT** illégal ou d'un **ARRÊT** illégal.

## Description des Registres TWI

L'interface **TWI** utilise 5 registres pour contrôler et gérer totalement l'interface.

### Registre **TWCR** (*TWI Control Register*)

Le registre **TWCR** est employé pour contrôler les opérations de l'interface. Il est employé pour permettre d'amorcer un accès au Maître en appliquant une condition **DÉBUT** sur le bus, placer l'interface en mode

récepteur (**ACK** ou **NACK**), produire une condition d'**ARRÊT** pour contrôler la halte du bus tandis que les données prêtes à être écrites sur le bus sont écrites dans le **TWDR**. Il indique aussi une collision d'écriture des données écrites dans **TWDR** tandis que le registre est inaccessible.

Adresse	7	6	5	4	3	2	1	0
\$36	<b>TWINT</b>	<b>TWEA</b>	<b>TWSTA</b>	<b>TWSTO</b>	<b>TWWC</b>	<b>TWEN</b>	-	<b>TWIE</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L	L/E

**TWINT TWI Interrupt Flag** Ce bit est mis à 1 par le matériel quand l'interface **TWI** a fini son travail actuel et attend la réponse du logiciel d'application. Si le bit **I** dans **SREG** et le bit **TWIE** dans **TWCR** est mis à 1, le **MCU** exécutera l'interruption du vecteur **TWI**. Tandis que le drapeau **TWINT** est mis à 1, l'horloge **SCL** est mise en bas pendant l'interruption. Le drapeau **TWINT** doit être mis à 0 par le logiciel, il n'est pas automatiquement remis à 0 par le matériel. Quand ce drapeau est mis à 0, l'opération de l'interface **TWI** commence, donc les registres **TWAR**, **TWSR** et **TWDR** doivent être configurés avant le dégagement du drapeau.

**TWEA TWI Enable Acknowledge Bit** Le bit contrôle la génération de l'impulsion de reconnaissance **ACK**. Si le bit **TWEA** est à 1, l'impulsion **ACK** est produite sur le bus si les conditions suivantes sont rencontrées :

1. La propre adresse d'Esclave du dispositif a été reçue.
2. Un appel général a été reçu, tandis que la bit **TWGCE** dans le **TWAR** est mis à 1.
3. Un octet de données a été reçu dans le Récepteur du Maître ou le Récepteur de l'Esclave.

En écrivant le bit **TWEA** à 0, le dispositif est considéré comme débranché temporairement du bus. La reprise se fait en écrivant le bit **TWEA** à 1 de nouveau.

**TWSTA TWI START Condition Bit** L'application écrit le bit à 1 quand il désire devenir Maître sur le bus. Le matériel vérifie que le bus est disponible et produit une condition de **DÉBUT**. Si le bus n'est pas libre, l'interface attend qu'une condition d'**ARRÊT** soit détectée pour produire ensuite une nouvelle condition de **DÉBUT** et revendique le statut de Maître du bus. **TWSTA** doit être mis à 0 par le logiciel quand la condition de **DÉBUT** a été transmise.

**TWSTO TWI STOP Condition Bit** L'écriture du bit à 1 dans en mode Maître produira une condition d'**ARRÊT** sur le bus. Quand la condition d'**ARRÊT** est exécutée sur le bus, le bit **TWSTO** est mis à 0 automatiquement. En mode Esclave, le bit est mis à 1 pour se remettre d'une condition d'erreur. Cela ne produira pas de condition d'**ARRÊT**, mais positionne l'interface dans un mode Esclave non adressé et remet les lignes **SCL** et **SDA** à l'état haut.

**TWWC TWI Write Collision Flag** Le bit est mis à 1 quand une tentative d'écriture dans le registre de données **TWDR** est réalisée et que le bit **TWINT** est à 0. Ce drapeau est mis à 0 en écrivant le registre de **TWDR** et que le bit **TWINT** est à 1.

**TWEN TWI Enable Bit** Le bit mis à 1 permet d'activer l'interface **TWI** et prend le contrôle des broches d'entrée-sortie connectées sur **SCL** et **SDA**, active les limiteurs de vitesse et les filtres de pointes. Si ce bit est écrit à 0, l'interface **TWI** est éteinte et toutes les transmissions **TWI** sont terminées, quelle que soit l'opération en cours.

**TWIE TWI Interrupt Enable** Quand ce bit est à 1 et que le bit **I** dans **SREG** est à 1, l'interface **TWI** active les demandes d'interruption quand le drapeau **TWINT** est placé à l'état haut.

### Registre **TWBR** (*TWI Bit Rate Register*)

Le registre **TWBR** permet de définir la fréquence de travail de l'interface.

Adresse	7	6	5	4	3	2	1	0
\$00	<b>TWBR7</b>	<b>TWBR6</b>	<b>TWBR5</b>	<b>TWBR4</b>	<b>TWBR3</b>	<b>TWBR2</b>	<b>TWBR1</b>	<b>TWBR0</b>
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**TWBR7:0 TWI bit Rate Register** Choisit le facteur de division pour le générateur de taux de bit. Le générateur de taux de bit est un diviseur de fréquence qui produit la fréquence d'horloge **SCL** dans les modes Maître. Voir "l'Unité de Générateur de Taux de Bit" avec **TWBR** > 10.



$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{\text{TWPS}}}$$

**TWPS** = Valeur du pré diviseur (1 à 64).

Par exemple pour une fréquence d'horloge MCU de 8 MHz et un pré diviseur de 1, la valeur 10 donnera une fréquence SCL de :  $8.000.000 / 16 + (2 \times 10) \times 4^1$  soit  $8.000.000 / 96 = 83\,333$  Hz soit environ 83 KHz.

### Registre TWSR (*TWI Status Register*)

Le registre de statut de l'interface TWI permet de connaître la position des éléments de l'interface.

Adresse	7	6	5	4	3	2	1	0
\$01	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0
L/E	L	L	L	L	L	L	L/E	L/E

**TWS7:3 TWI Status 7 à 3** Ces cinq bits reflètent le statut de la logique de l'interface TWI et du bus. Les codes de statut sont décrits un peu plus tard dans ce paragraphe. La valeur lue contient le statut sur 5 bits et le pré diviseur sur 2 bits. L'application doit masquer les bits du pré diviseur lors de la vérification du statut.

**TWPS1:0 TWI Prescaler Bits** Ces bits peuvent être lus et écrits pour contrôler le taux de transfert du pré diviseur. Pour calculer des taux de transfert, voir "*l'Unité de Générateur de Taux de Bit*" ou les valeur de TWPS1:0 sont employé dans l'équation.

TWPS1	TWPS0	Valeur du Pré-diviseur
0	0	1
0	1	4
1	0	16
1	1	64

### Registre TWAR (*TWI (Slave) Address Register*)

Le registre TWAR doit être chargé de l'adresse de l'Esclave à 7 bits auquel le TWI répondra en émetteur d'adresse Esclave ou en récepteur. Dans le mode Multi Maître, TWAR doit être rempli pour être adressé comme un Esclave par d'autres Maîtres.

Adresse	7	6	5	4	3	2	1	0
\$02	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E

**TWA6:0 TWI(Slave) Address 6 à 0 Register** Ces sept bits constituent l'adresse Esclave de l'unité TWI.

**TWGCE TWI General Call Recognition Enable Bit** Si mis à 1, ce bit permet l'identification d'un appel général réservé sur le bus, si mis à 0, l'appel général est ignoré.

### Registre TWDR (*TWI Data Register*)

Dans le mode de Transmission, TWDR contient le prochain l'octet à être transmis. Dans le mode Réception, TWDR contient le dernier octet reçu. L'octet est présent quant l'interface n'est pas dans le processus de chargement d'un octet, soit quand le drapeau TWINT est mis à 1 par le matériel. Le registre de donnée ne peut pas être initialisé par l'utilisateur avant que la première interruption arrive.

L'octet de donnée reste stable tant que TWINT est à 1. Quant le transfert de donnée est cours sur le bus, la donnée est simultanément chargée dans TWDR, donc le registre contient toujours le dernier octet du bus, sauf lors de la sortie du mode sommeil. Dans ce cas, le contenu de TWDR est non défini.

Dans le cas d'un arbitrage de bus perdu, aucune donnée n'est perdue dans la transition Maître vers Esclave. Le traitement du bit ACK est contrôlé automatiquement par la logique de l'interface. Le programme ne peut pas avoir accès directement au bit ACK.

Adresse	7	6	5	4	3	2	1	0
\$03	TWDR7	TWDR6	TWDR5	TWDR4	TWDR3	TWDR2	TWDR1	TWDR0

L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
-----	-----	-----	-----	-----	-----	-----	-----	-----

**TWD7:0 TWI Data 7 à 0 Register** Le prochain octet de données à être transmis, ou le dernier octet de données reçu sur le bus.

## Code des Statut TWSR

### Mode Transmission Maître avec TWINT = 1

Statut TWSR	Statut	Réponse sur TWDR	STA	STO	Action suivante
\$08	Début	SLA+W	0	0	Adresse sur le bus, <b>ACK</b> ou <b>NACK</b> à recevoir
\$10	Début Répété	SLA+W SLA+R	0	0	SLA+W transmit : <b>ACK</b> ou <b>NACK</b> reçu SLA+R transmit : Réception en Maître
\$18	SLA+W avec ACK	Donnée ok Rien Rien Rien	0 1 0 1	0 0 1 1	Donnée à transmettre sur le bus Début Répété Arrêt et raz <b>TWSTO</b> Arrêt suivi d'un Début et raz <b>TWSTO</b>
\$20	SLA+W avec NACK	Donnée ok Rien Rien Rien	0 1 0 1	0 0 1 1	Donnée sur le bus et <b>ACK</b> ou <b>NACK</b> à recevoir Début Répété Arrêt et raz <b>TWSTO</b> Arrêt suivi d'un Début et raz <b>TWSTO</b>
\$28	Donnée avec ACK	Donnée ok Rien Rien Rien	0 1 0 1	0 0 1 1	Donnée sur le bus et <b>ACK</b> ou <b>NACK</b> à recevoir Début Répété Arrêt et raz <b>TWSTO</b> Arrêt suivi d'un Début et raz <b>TWSTO</b>
\$30	Donnée avec NACK	Donnée ok Rien Rien Rien	0 1 0 1	0 0 1 1	Donnée sur le bus et <b>ACK</b> ou <b>NACK</b> à recevoir Début Répété Arrêt et raz <b>TWSTO</b> Arrêt suivi d'un Début et raz <b>TWSTO</b>
\$38	Arbitrage	Rien Rien	0 1	0 0	Passé en mode Esclave, perte du Maître Attend la libération du bus et envoie un Début

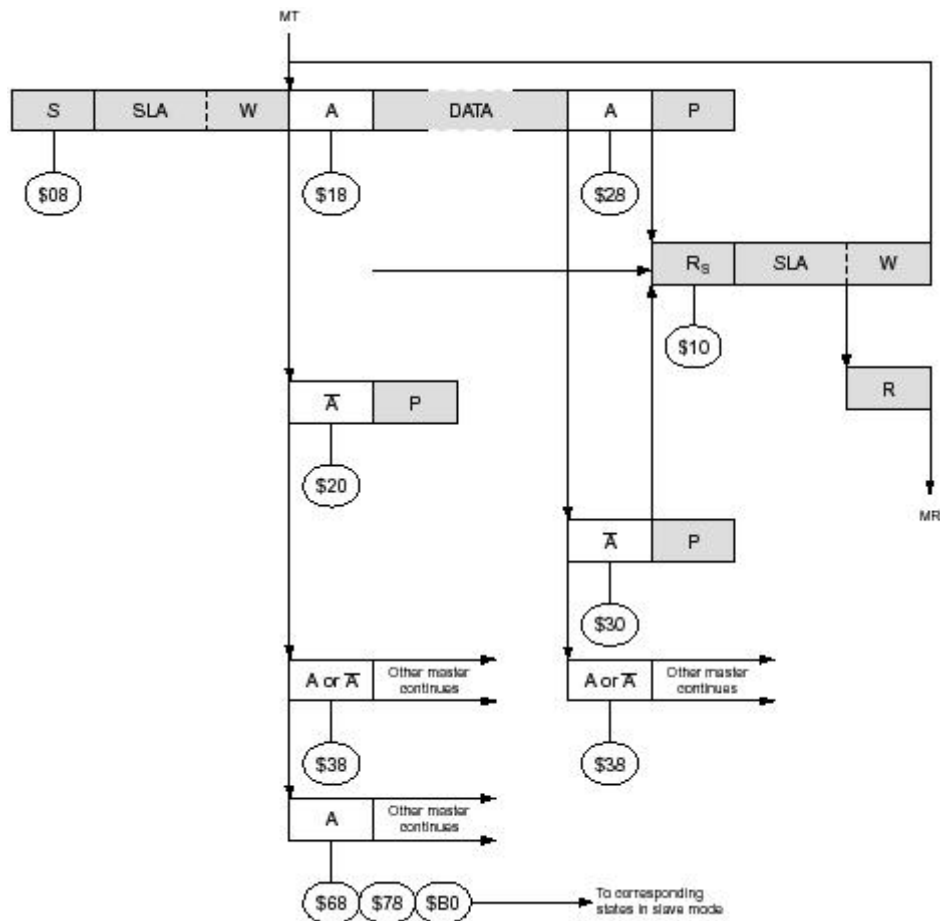


Figure 52, Code de statut en Transmission Maître.

### Mode Réception Maître WTINT = 1

Statut TWSR	Statut	Réponse sur TWDR	STA	STO	TWEA	Action suivante
\$08	Début	SLA+W	0	0	X	Adresse sur le bus, ACK ou NACK à recevoir
\$10	Début Répété	SLA+W SLA+R	0	0	X	SLA+W transmit : ACK ou NACK reçu SLA+R transmit : Réception en Maître
\$38	Arbitrage SLA+R ou NACK	Rien	0	0	X	Adresse inexistant ou Bus collision, bascule en mode Escale
		Rien	1	0	X	Début transit quant bus libre
\$40	SLA+R avec ACK	Rien	0	0	0	Donnée reçue et NACK retourné
		Rien	0	0	1	Donnée reçue et ACK retourné
\$48	SLA+R avec NACK	Rien	1	0	X	Début Répété à transmettre
		Rien	0	0	X	Arrêt et raz TWSTO
			1	1	X	Arrêt suivi d'un Début et raz TWSTO
\$50	Donnée reçue ACK retourné	Lire	0	0	0	Donnée reçue et NACK retourné
		Donnée	1	0	1	Donnée reçue et ACK retourné
\$58	Donnée reçue NACK retourné	Lire	1	0	X	Début Répété à transmettre
		Donnée	0	1	X	Arrêt et raz TWSTO
			1	1	X	Arrêt suivi d'un Début et raz TWSTO

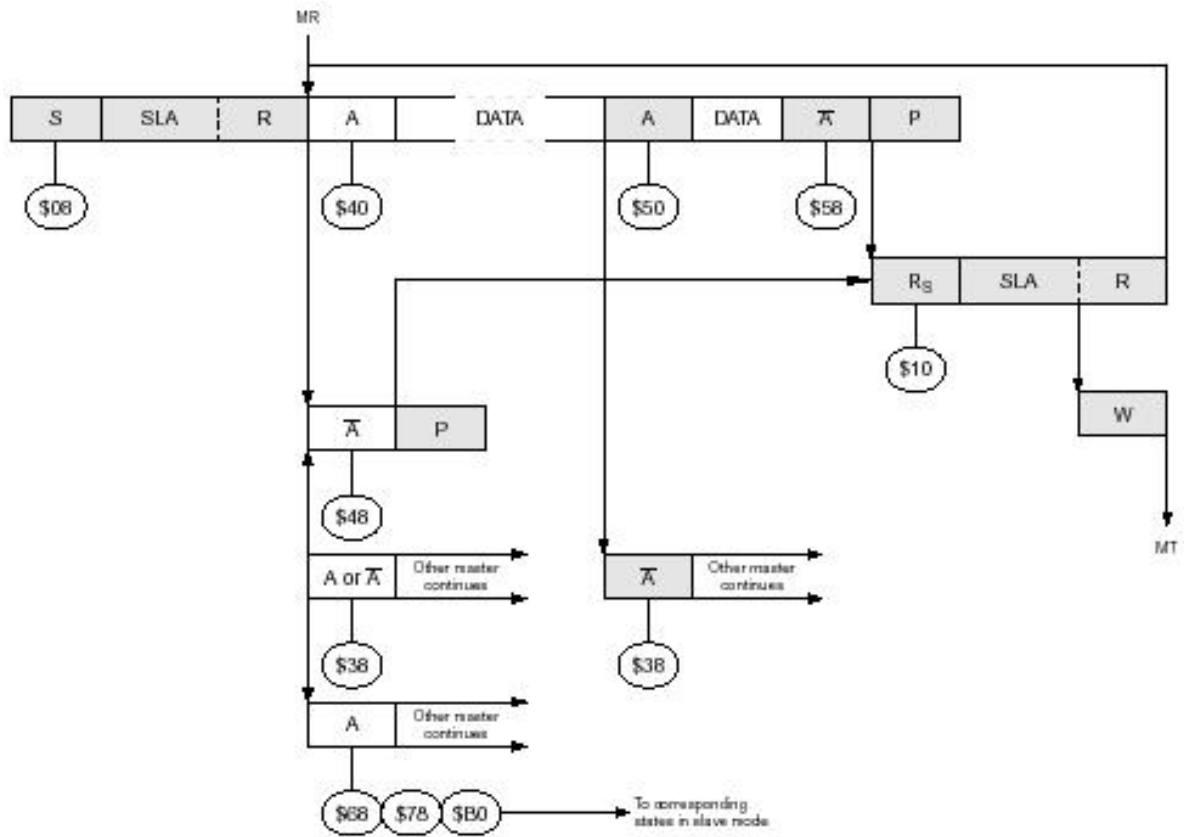


Figure 53, Code statut en Réception Maître.

### Mode Réception Esclave WTINT = 1

Statut TWSR	Statut	Réponse sur TWDR	STA	STO	TWEA	Action suivante
\$60	SLA+W reçue ACK retourné	Rien	X	0	0	Donnée reçue et <b>NACK</b> retourné
		Rien	X	0	1	Donnée reçue et <b>ACK</b> retourné
\$68	Arbitrage sur SLA+R/W ACK retourné	Rien	X	0	0	Donnée reçue et <b>NACK</b> retourné
		Rien	X	0	1	Donnée reçue et <b>ACK</b> retourné
\$70	Adresse Général reçue ACK retourné	Rien	X	0	0	Donnée reçue et <b>NACK</b> retourné
		Rien	X	0	1	Donnée reçue et <b>ACK</b> retourné
\$78	Arbitrage+ Adresse Général reçue ACK retourné	Rien	X	0	0	Donnée reçue et <b>NACK</b> retourné
		Rien	X	0	1	Donnée reçue et <b>ACK</b> retourné
\$80	SLA+W ok avec ACK	Lire	0	0	0	Donnée reçue et <b>NACK</b> retourné
		Donnée	1	0	1	Donnée reçue et <b>ACK</b> retourné
\$88	SLA+W ok avec NACK	Lire	0	0	0	Commute en Esclave non adressé sans SLA
		Donnée	0	0	1	Commute en Esclave non adressé avec SLA
			1	0	0	Commute en Esclave non adressé sans SLA + Début transit quant bus libre
			1	0	1	Commute en Esclave non adressé avec SLA + Début transit quant bus libre
\$90	Adresse Général reçue ACK retourné	Lire	X	1	0	Donnée reçue et <b>NACK</b> retourné
		Donnée	X	1	1	Donnée reçue et <b>ACK</b> retourné
\$98	Adresse Général reçue NACK retourné	Lire	0	0	0	Commute en Esclave non adressé sans SLA
		Donnée	0	0	1	Commute en Esclave non adressé avec SLA
			1	0	0	Commute en Esclave non adressé sans SLA + Début transit quant bus libre
			1	0	1	Commute en Esclave non adressé avec SLA + Début transit quant bus libre
\$A0	Arrêt ou Début Répété reçu avec SLA Ok	Rien	0	0	0	Commute en Esclave non adressé sans SLA
			0	0	1	Commute en Esclave non adressé avec SLA
			1	0	0	Commute en Esclave non adressé sans SLA + Début transit quant bus libre
			1	0	1	Commute en Esclave non adressé avec SLA + Début transit quant bus libre

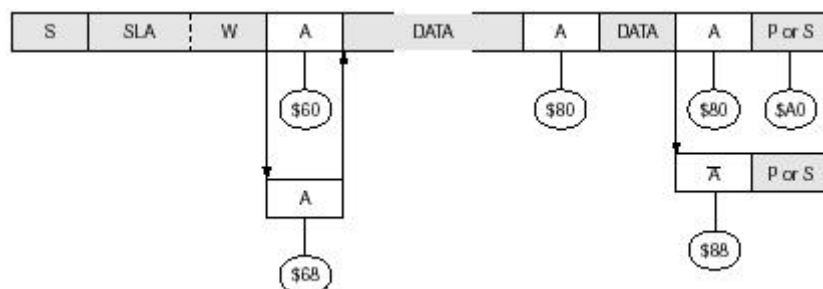


Figure 54, Code de statut en réception Esclave.

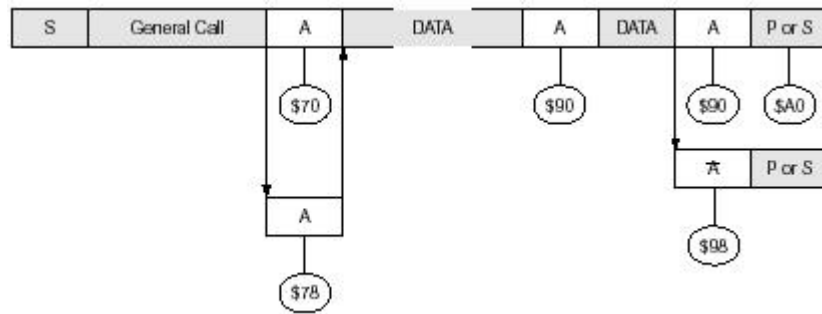


Figure 55, Code de statut en réception d'adresse générale Esclave.

### Mode Transmission Esclave WTINT = 1

Statut TWSR	Statut	Réponse sur TWDR	STA	STO	TWEA	Action suivante
\$A8	SLA+R reçue ACK retourné	Lire Donnée	X X	0 0	0 1	Donnée transmise et <b>NACK</b> retourné Donnée transmise et <b>ACK</b> retourné
\$B0	Arbitrage sur SLA+R ACK retourné	Lire Donnée	X X	0 0	0 1	Donnée transmise et <b>NACK</b> retourné Donnée transmise et <b>ACK</b> retourné
\$B8	Donnée TWDR transmis ACK retourné	Lire Donnée	X X	0 0	0 1	Donnée transmise et <b>NACK</b> retourné Donnée transmise et <b>ACK</b> retourné
\$C0	Donnée TWDR transmis NACK retourné	Rien	0 0 1 1	0 0 0 0	0 1 0 1	Commute en Esclave non adressé sans <b>SLA</b> Commute en Esclave non adressé avec <b>SLA</b> Commute en Esclave non adressé sans <b>SLA</b> + Début transit quant bus libre Commute en Esclave non adressé avec <b>SLA</b> + Début transit quant bus libre
\$C8	Donnée TWDR transmis ACK retourné	Rien	0 0 1 1	0 0 0 0	0 1 0 1	Commute en Esclave non adressé sans <b>SLA</b> Commute en Esclave non adressé avec <b>SLA</b> Commute en Esclave non adressé sans <b>SLA</b> + Début transit quant bus libre Commute en Esclave non adressé avec <b>SLA</b> + Début transit quant bus libre

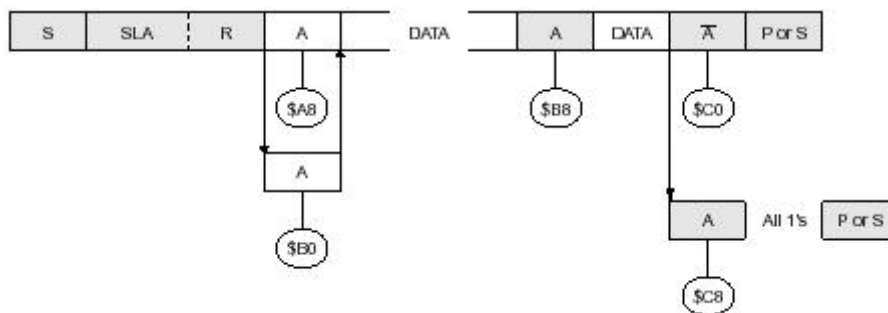


Figure 56, Code Statut Transmission Esclave.

**Symbole :** En bleu clair Maître vers Esclave, en blanc Esclave vers Maître, Rond = N° Statut.

### Code de Statut Indéfini

Statut TWSR	Statut	TWDR	STA	STO	TWEA	Action suivante
\$F8	TWINT = 0	Rien				TWINT = 0, pas d'action
\$00	Début ou Arrêt illégale	Rien	0	1	X	Raz TWSTO

## Programmation de l'Interface TWI

L'interface **TWI** est basée sur le transfert d'octet par interruption. Les interruptions sont publiées après tous les événements du bus, comme la réception d'un octet ou la transmission d'une condition de **DÉBUT**. Parce que l'interface **TWI** est basée sur les interruptions, le logiciel d'application est libre de réaliser d'autres opérations pendant un transfert d'octet **TWI**. La permission de l'interruption se fait avec le bit **TWIE** dans **TWCR** et l'activation de la gestion des interruptions générale dans **SREG**. L'application décide en fonction du drapeau **TWINT** si une demande d'interruption doit être validée.

Si le bit **TWIE** est mis à 0, l'application doit lever le drapeau **TWINT** pour détecter des actions du bus. Quand le drapeau **TWINT** est à 1, l'interface **TWI** a fini une opération et attend la réponse de l'application. Dans ce cas, le registre de statut **TWSR** contient une valeur indiquant l'état actuel du bus. Le logiciel d'application peut alors décider du traitement à faire pour piloter le comportement du bus dans le cycle suivant en manipulant des registres **TWDR** et **TWCR**.

### Paramétrage de l'interface TWI

La figure qui suit donne un exemple simple du fonctionnement l'application avec l'interface matérielle **TWI**. Dans cet exemple, un Maître veut transmettre un octet de donnée simple à un Esclave. Cette description est tout à fait abstraite, une explication plus détaillée suit dans ce chapitre. Un exemple de code simple mettant en oeuvre le comportement désirer est aussi présenté. Après l'initialisation de l'interface **TWI** avec l'activation de l'interruption générale et celle de l'interface, les étapes sont les suivantes pour une transmission :

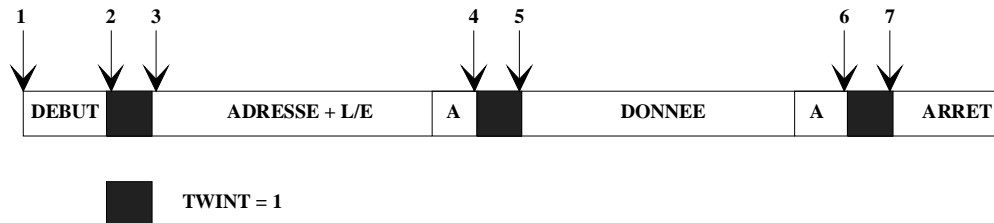


Figure 57, Exemple de fonctionnement **TWI** simple.

1. En premier intervient une écriture dans **TWCR** pour transmettre une condition de **DÉBUT** sur le bus. C'est fait en écrivant le bit **TWSTA** à 1 dans **TWCR**. L'interface regardera l'état du bus et seulement si il est disponible, à l'état haut, il transmet la condition, si le bus est occupé, la condition sera écrit plus tard qu'une condition d'**ARRÊT** soit transmit. Le bit **TWINT** est positionné à 1 sur l'écriture sur le bus et sera remis à 0 à la fin de l'écriture. L'interface **TWI** commencera l'opération quant le bit **TWINT** est à 0.
2. Quand la condition **DÉBUT** a été transmise, le drapeau **TWINT** est mis à 1 et **TWSR** est mis à jour avec un code statut indiquant que la condition **DÉBUT** a été envoyée avec succès.
3. Le logiciel d'application doit maintenant examiner la valeur de **TWSR**, s'assurer que la condition de **DÉBUT** a été transmise avec succès. Si **TWSR** indique autre chose, le logiciel d'application pourrait appeler une routine d'erreur. Si tout est correct, la valeur d'adresse **SLA+W** est placé dans **TWDR** puis le bit **TWINT** et **TWEN** sont mis à 1 dans **TWCR**, indiquant à l'interface de transmettre l'adresse dans **TWDR**. L'adresse sera transmise plus tard sur le bus quant le bit **TWINT** sera mis à 0 par l'interruption. Immédiatement après l'interface amorcera la transmission du paquet d'adresse sur le bus.
4. Quand le paquet d'adresse a été transmis, le drapeau **TWINT** mis à 1 et **TWSR** est mis à jour avec le code du statut 'paquet d'adresse envoyé avec succès'. Le code du statut reflétera aussi la reconnaissance d'un Esclave qui a reconnu ou non l'adresse.
5. Le logiciel d'application doit maintenant examiner la valeur de **TWSR**, s'assurer que le paquet d'adresse a été transmis avec succès et que l'on a la valeur **ACK**. Si **TWSR** indique autre chose, le logiciel d'application pourrait appeler une routine d'erreur par exemple. Le paquet de données est ensuite mis dans **TWDR** et puis le bit **TWINT** et **TWEN** sont mis à 1 dans **TWCR**, indiquant à l'interface de transmettre la donnée contenue dans **TWDR**. La donnée sera transmise plus tard sur

le bus quant le bit **TWINT** sera mis à 0 par l'interruption. Immédiatement après l'interface amorcera la transmission du paquet de donnée sur le bus.

6. Quand le paquet de données a été transmis, le drapeau **TWINT** est mis à 1 et **TWSR** est mis à jour avec le code de statut indiquant que le paquet de données a été envoyé avec succès. Le code de statut reflétera aussi si un Esclave a reconnu le paquet ou non.
7. Le logiciel d'application doit maintenant examiner la valeur de **TWSR** et s'assurer que le paquet de données a été transmis avec succès et que l'on a la valeur **ACK**. Si **TWSR** indique autre chose, le logiciel d'application pourrait appeler une routine d'erreur. Et pour finir le bit **RWST** est écrit à 1 dans **TWCR** pour transmettre une condition d'**ARRÊT**. **TWINT** n'est pas mis à 1 après une condition d'**ARRÊT**.

Bien que cet exemple soit simple, il montre le principe appliqué dans toutes les transmissions **TWI**.

### Exemple de programmation

Dans l'exemple qui suit, un programme en assembleur reprend la démonstration :

```

Etape1:                                ; Préparation de la transmission TWI
      Ldi    r16, (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)
      Out    TWCR, r16                ; Envoie la condition DEBUT
Etape2:                                ; Attendre TWINT = 1 pour Début transmis
wait1:  in    r16, TWCR                ; Charge le control TWI
      sbrs   r16, TWINT              ; Test du bit TWINT
      rjmp   wait1                   ; Attendre TWINT=1 pour DEBUT transmis
Etape3:                                ; Test le statut de transmission ok/Nok
      In     r16, TWSR                ; Charge le statut TWI
      Andi   r16, 0xF8                ; Masque le pré diviseur
      Cpi    r16, START               ; Comparaison avec valeur OK
      Brne   ERROR                   ; Branchement si erreur
      Ldi    r16, SLA_W               ; Chargement de l' Adresse
      Out    TWDR, r16               ; Place l' adresse dans TWDR
      Ldi    r16, (1<<TWINT) | (1<<TWEN)
      Out    TWCR, r16                ; Active la transmission sur le Bus
Etape4:                                ; SLA+W a été transmis et ACK/NACK a été reçu
wait2:  ; Attendre TWINT = 1
      in     r16, TWCR                ; Charge le statut
      sbrs   r16, TWINT              ; Test TWINT = 1
      rjmp   wait2                   ; Boucle si = 0
Etape5:                                ; DONNÉES de charge pour la transmission de données
      In     r16, TWSR                ; Charge le statut
      Andi   r16, 0xF8                ; Masque le pré diviseur
      Cpi    r16, MT_SLA_ACK          ; Compare avec Ok ou Erreur
      Brne   ERROR                   ; Branchement sur erreur
      ; Vérifiez le Statut et test si ok ou en erreur si ok donnée
      Ldi    r16, DATA               ; Charge la donnée
      Out    TWDR, r16               ; Sort la donnée en transmission
      Ldi    r16, (1<<TWINT) | (1<<TWEN)
      Out    TWCR, r16                ; Active la transmission
Etape6:                                ; Attendre TWINT=1. DONNÉES transmises
wait3:  in     r16, TWCR                ; et ACK/NACK a été reçu.
      sbrs   r16, TWINT              ; Test TWINT=1
      rjmp   wait3                   ; Boucle si pas finie
Etape7:                                ; Transmission Donnée ok ou nok et fin
      In     r16, TWSR                ; Charge le statut
      Andi   r16, 0xF8                ; Masque le pré diviseur
      Cpi    r16, MT_DATA_ACK         ; Vérifiez le Statut Si statut on ou nok
      Brne   ERROR                   ; Branchement si erreur
      Ldi    r16, (1<<TWINT)|(1<<TWEN)|(1<<TWSTO)
      Out    TWCR, r16                ; Transmet la condition d' ARRET

```



# Verrouillage des Programmes et des Données

L'ATMEGA32 fournit six bits de verrouillage qui peuvent être laissées non programmées à 1 où peuvent être programmées à 0 pour obtenir les particularités complémentaires inscrites dans la table suivante.

**Attention :** Les bits de verrouillage ne peuvent être écrit qu'à 1 avec la commande d'effacement du circuit.

Verrouillage	No	Valeur par Défaut	Description
-	7	-	1 (non programmé)
-	6	-	1 (non programmé)
BLB12	5	Verrouillage Boot	1 (non programmée)
BLB11	4	Verrouillage Boot	1 (non programmée)
BLB02	3	Verrouillage Boot	1 (non programmée)
BLB01	2	Verrouillage Boot	1 (non programmée)
LB2	1	Verrouillage	1 (non programmée)
LB1	0	Verrouillage	1 (non programmée)

**Note :** 1 = Non programmé, 0 = programmé (non modifiable).

## Verrouillage bit Mémoire (2)

Mode	LB2	LB1	Type de Protection
1	1	1	Pas de verrouillage activé, lecture et écriture possible.
2	1	0	Le blocage de la programmation de la mémoire <b>FLASH</b> et <b>EEPROM</b> est mis en service dans le mode de Programmation Parallèle et <b>SPI/JTAG</b> . Il est impossible d'écrire dans les mémoires (1)
3	0	0	Le blocage de la programmation et de la vérification de la <b>FLASH</b> et <b>EEPROM</b> est mise en service dans le mode de Programmation Parallèle et <b>SPI/JTAG</b> . Il est impossible de lire et d'écrire dans les mémoires.(1)
Mode	BLB02	BLB01	Type de Protection
1	1	1	Aucune restriction pour <b>SPM</b> ou <b>LPM</b> , accès à la section d'application.
2	1	0	<b>SPM</b> ne permettent pas d'écrire à la section d'application.
3	0	0	<b>SPM</b> ne permettent pas pour d'écrire à la section d'application et on ne permet pas à <b>LPM</b> exécuter en mode boot de lire de la section d'application. Si les vecteurs d'interruption sont placés dans la section de boot, les interruptions sont mises hors service en exécutant la section d'application.
4	0	1	<b>LPM</b> exécuter en mode boot ne permet pas de lire de la section d'application. Les vecteurs d'interruption sont placés dans la section de boot, les interruptions sont mis hors de service en exécutant la section d'application.
Mode	BLB12	BLB11	Type de Protection
1	1	1	Aucune restriction pour <b>SPM</b> ou <b>LPM</b> accès à la section boot.
2	1	0	<b>SPM</b> ne permettent pas d'écrire à la section boot.
3	0	0	<b>SPM</b> ne permettent pas d'écrire à la section boot et on ne permet pas à <b>LPM</b> exécutant la section d'application de lire de la section boot. Si les vecteurs d'interruption sont placés dans la section d'application, les interruptions sont mises hors de service en exécutant de la section de boot.
4	0	1	<b>LPM</b> exécutant la section d'application ne permettent pas de lire la section de boot. les vecteurs d'interruption sont placés dans la section d'application, les interruptions sont mises hors de service en exécutant la section boot.

**Notes :** 1. Programmer les bits de fusible avant de programmer les bits de verrouillage.

2. "1" = non programmés, "0" = programmés et non modifiable.

## Fusible de Programmation

L'ATMEGA32 a deux octets de fusible. Les deux tables qui suivent décrivent brièvement la fonctionnalité de tous les fusibles et comment ils sont initialisés.

### Bit de Fusible haut

Fusible haut	No	Description	Valeur par défaut
OC DEN <sup>(4)</sup>	7	Permet <b>OCD</b> (Debug)	1 (non programmé, <b>OCD</b> hors service)
JTAG EN <sup>(5)</sup>	6	Permet <b>JTAG</b>	0 (programmé, <b>JTAG</b> permis)
SPI EN <sup>(1)</sup>	5	Permet le Téléchargement des Programmes par l'interface <b>SPI</b>	0 (programmé, <b>SPI</b> Prog. permis)
CKOPT <sup>(2)</sup>	4	Options d'Oscillateur	1 (non programmé)
EESAVE	3	<b>EEPROM</b> préservé des Effacements	1 (non programmé, <b>EEPROM</b> non préservé)
BOOTSZ1	2	Taille du Boot (voir Table suivante)	0 (a programmé) <sup>(3)</sup>
BOOTSZ0	1	Taille du Boot (voir Table suivante)	0 (a programmé) <sup>(3)</sup>
BOOTRST	0	Sélection du vecteur de <b>RESET</b>	1 (non programmé)

Notes 1 : Le Fusible SPIEN n'est pas accessible dans le mode de Programmation **SPI**.

2. La fonctionnalité de fusible de **CKOPT** dépend des bits **CKSEL**. Voir "Sources d'Horloge" au début.
3. La valeur par défaut de **BOOTSZ1:0** est la Taille du programme de 'boot' maximale. Voir la Table qui suit.
4. Mettre le bit **OC DEN** à 1 quand le programme est terminé (arrêt du mode debug) Un Fusible **OC DEN** programmé permet de modifier l'horloge système pour tester tous les modes sommeil. Cela peut augmenter la consommation.
5. Si l'interface **JTAG** n'est pas utilisé, le fusible **JTAG EN** doit si possible d'être mis hors de service. Cela pour éviter les courants statiques sur la broche **TDO** dans l'interface **JTAG**.

### Bit de Fusible bas

Le statut des bits de fusibles n'est pas affecté par l'effacement du composant. Les bits de fusible sont verrouillés si **LB1** est programmé. Programmer les bits de fusible avant programmation des bits de verrouillage.

Fusible Bas	No	Description	Valeur par Défaut
BODLEVEL	7	Bascule du Détecteur de Panne d'électricité	1 (non programmé)
BODEN	6	Active Détecteur de Panne d'électricité	1 (non programmé, BOD hors service)
SUT1	5	Temps de démarrage	1 (non programmé) <sup>(1)</sup>
SUT0	4	Temps de démarrage	0 (programmé) <sup>(1)</sup>
CKSEL3	3	Source d'Horloge	0 (programmé) <sup>(2)</sup>
CKSEL2	2	Source d'Horloge	0 (programmé) <sup>(2)</sup>
CKSEL1	1	Source d'Horloge	0 (programmé) <sup>(2)</sup>
CKSEL0	0	Source d'Horloge	1 (non programmé) <sup>(2)</sup>

Notes 1 : La valeur de défaut de **SUT1:0** est le résultats du temps de démarrage maximal.

2. Les bits **CKSEL3:0** sélectionne la source d'horloge de l'oscillateur interne **RC 1 MHz**.

# Jeu d’Instruction

L’ATMEGA à un jeu de 131 instructions qui seront détaillées dans le deuxième document sur l’assembleur ATMEGA. Elles sont présentées dans les tableaux qui suivent :

Code	Opérant	Description	Opération	Drapeau	Cycle
<b>Instructions Arithmétiques et Logique</b>					
ADD	Rd, Rr	Addition sans Retenue	$Rd = Rd + Rr$	Z C N V S H	1
ADC	Rd, Rr	Addition avec Retenue	$Rd = Rd + Rr + C$	Z C N V S H	1
ADIW	Rd, K	Addition Immédiate 16 bits	$Rd+1:Rd = Rd+1:Rd + K$	Z C N V S	2 (1)
SUB	Rd, Rr	Soustraction sans Retenue	$Rd = Rd - Rr$	Z C N V S H	1
SUBI	Rd, K	Soustraction Immédiat sans Retenue	$Rd = Rd - K$	Z C N V S H	1
SBC	Rd, Rr	Soustraction avec Retenue	$Rd = Rd - Rr - C$	Z C N V S H	1
SBCI	Rd, K	Soustraction Immédiat avec Retenue	$Rd = Rd - K - C$	Z C N V S H	1
SBIW	Rd, K	Soustraction Immédiat 16 bits	$Rd+1:Rd = Rd+1:Rd - K$	Z C N V S	2 (1)
ET	Rd, Rr	<b>ET</b> Logique	$Rd = Rd \& Rr$	Z N V S	1
ANDI	Rd, K	<b>ET</b> Logique Immédiat	$Rd = Rd \& K$	Z N V S	1
OU	Rd, Rr	<b>OU</b> Logique	$Rd = Rd ! Rr$	Z N V S	1
ORI	Rd, K	<b>OU</b> Logique Immédiat	$Rd = Rd ! K$	Z N V S	1
EOR	Rd, Rr	<b>OU</b> Exclusif	$Rd = Rd \oplus Rr$	Z N V S	1
COM	Rd	Complément à 1	$Rd = \$FF - Rd$	Z C N V S	1
NEG	Rd	Négation (Complément à 2)	$Rd = \$00 - Rd$	Z C N V S	1
SBR	Rd, K	Mise à 1 dans Registre ( <b>OU</b> )	$Rd = Rd ! K$	Z N V S	1
CBR	Rd, K	Mise à 0 dans Registre ( <b>ET</b> )	$Rd = Rd \& (\$FF - K)$	Z N V S	1
INC	Rd	Incrément	$Rd = Rd + 1$	Z N V S	1
DEC	Rd	Décrément	$Rd = Rd - 1$	Z N V S	1
TST	Rd	Test à Zéro ou Négatif	$Rd = Rd ! Rd$	Z N V S	1
CLR	Rd	Effacement du Registre	$Rd = \$00$	Z N V S	1
SER	Rd	Mis à 1 du Registre	$Rd = \$FF$	-	1
MUL	Rd, Rr	Multiplication non Signé	$R1:R0 = Rd \times Rr$ (UU)	Z C	2 (1)
MULS	Rd, Rr	Multiplication Signé	$R1:R0 = Rd \times Rr$ (SS)	Z C	2 (1)
MULSU	Rd, Rr	Multiplication Signé avec non Signé	$R1:R0 = Rd \times Rr$ (SU)	Z C	2 (1)
FMUL	Rd, Rr	Multiplication Fractionnaire non Signé	$R1:R0=(Rd \times Rr)\ll 1$ (UU)	Z C	2 (1)
FMULS	Rd, Rr	Multiplication Fractionnaire Signé	$R1:R0=(Rd \times Rr)\ll 1$ (SS)	Z C	2 (1)
FMULSU	Rd, Rr	Multiplication Fractionnaire Signé avec non Signé	$R1:R0=(Rd \times Rr)\ll 1$ (SU)	Z C	2 (1)

Code	Opérant	Description	Opération	Drapeau	Cycle
<b>Instructions de Branchement et Saut</b>					
RJMP	k	Saut Relatif	$PC = PC + k + 1$	-	2
IJMP		Saut Indirect Z	$PC(15:0)=Z, PC(21:16)=0$	-	2 (1)
EIJMP		Saut Etendu Indirect Z	$PC(15:0)=Z,$ $PC(21:16)=EIND$	-	2 (1)
JMP	k	Saut	$PC = k$	-	3 (1)
RCALL	k	Sous-Programme Relatif	$PC = PC + k + 1$	-	3/4 (4)
ICALL		Sous-Programme Indirect Z	$PC(15:0)=Z, PC(21:16)=0$	-	3/4(1,4)
EICAL L		Sous-Programme Etendu Indirect Z	$PC(15:0)=Z,$ $PC(21:16)=EIND$	-	4 (4)
CALL	k	Sous-Programme	$PC = k$	-	4/5(1,4)
RET		Retour de Sous-Programme	$PC = STACK$	-	4/5 (4)
RETI		Retour d'Interruption	$PC = STACK$	I	4/5 (4)
CPSE	Rd, Rr	Compare et Saute si Égal	Si $Rd=Rr$ $PC=PC+2$ ou 3	-	1/2/3
CP	Rd, Rr	Comparer	$Rd - Rr$	Z C N V S H	1
CPC	Rd, Rr	Comparez avec Retenue	$Rd - Rr - C$	Z C N V S H	1
CPI	Rd, K	Comparez Immédiat	$Rd - K$	Z C N V S H	1
SBRC	Rr, b	Sautez si le bit du Registre à 0	Si $Rr(b)=0$ $PC=PC+2$ ou 3	-	1/2/3
SBRS	Rr, b	Sautez si le bit du Registre à 1	Si $Rr(b)=1$ $PC=PC+2$ ou 3	-	1/2/3
SBIC	A, b	Sautez si le bit du Registre I/O à 0	Si $A, b=0$ $PC=PC+2$ ou 3	-	1/2/3
SBIS	A, b	Sautez si le bit du Registre I/O à 1	Si $A, b = 1$ $PC=PC+2$ ou 3	-	1/2/3
BRBS	s, k	Branche si <b>SREG(s)</b> à 1	Si $SREG(s)=1$ $PC=PC+k+1$	-	1/2
BRBC	s, k	Branche si <b>SREG(s)</b> à 0	Si $SREG(s)=0$ $PC=PC+k+1$	-	1/2
BREQ	K	Branche si Égal Z à 1	Si $Z=1$ $PC=PC+k+1$	-	1/2
BRNE	K	Branche si Non Égal Z à 0	Si $Z=0$ $PC=PC+k+1$	-	1/2
BRCS	K	Branche si Retenue C à 1	Si $C=1$ $PC=PC+k+1$	-	1/2
BRCC	K	Branche si Retenue C à 0	Si $C = 0$ $PC=PC+k+1$	-	1/2
BRSH	K	Branche si Egal ou Plus Haut	Si $C=0$ $PC=PC+k+1$	-	1/2
BRLO	K	Branche si Plus bas	si $C = 1$ $PC=PC+k+1$	-	1/2
BRMI	K	Branche si Négatif	Si $N=0$ $PC=PC+k+1$	-	1/2
BRPL	K	Branche si Positif	Si $N = 0$ $PC=PC+k+1$	-	1/2
BRGE	K	Branche si Plus Grand ou Égal Signé	Si $N\oplus V=0$ $PC=PC+k+1$	-	1/2
BRLT	K	Branche si Moins Que Signé	Si $N\oplus V=1$ $PC=PC+k+1$	-	1/2
BRHS	K	Branche si Drapeau Moitié H à 1	Si $H=1$ $PC=PC+k+1$	-	1/2
BRHC	K	Branche si Drapeau Moitié H à 0	Si $H=0$ $PC=PC+k+1$	-	1/2
BRTS	K	Branche si Drapeau T=1	Si $T=1$ $PC=PC+k+1$	-	1/2
BRTC	K	Branche si Drapeau T=0	Si $T=0$ $PC=PC+k+1$	-	1/2
BRVS	K	Branche si Débordement V=1	Si $V=1$ $PC=PC+k+1$	-	1/2
BRVC	K	Branche si Débordement v=0	Si $V=0$ $PC=PC+k+1$	-	1/2
BRIE	K	Branche si Interruption active I=1	Si $I=1$ $PC=PC+k+1$	-	1/2
BRID	K	Branche si Interruption inactive I=0	Si $I=0$ $PC=PC+k+1$	-	1/2

Code	Opérant	Description	Opération	Drapeau	Cycle
<b>Instructions de Transfert de Donnée</b>					
MOV	Rd, Rr	Déplacement	$Rd = Rr$	-	1
MOVW	Rd, Rr	Déplacement 16 bits	$Rd+1:Rd = Rr+1:Rr$	-	1 (1)
LDI	Rd, K	Charge Immédiat	$Rd = K$	-	1
LDS	Rd, k	Charge Directe en Mémoire	$Rd = (k)$	-	2 (1,4)
LD	Rd, X	Charge Indirect	$Rd = (X)$	-	2 (1,4)
LD	Rd, X+	Charge Indirect Post Incrément	$Rd = (X)$ et $X = X + 1$	-	2 (1,4)
LD	Rd, -X	Charge Indirect Pré Décrément	$X = X - 1$ et $Rd = (X)$	-	2 (1,4)
LD	Rd, Y	Charge Indirect	$Rd = (Y)$	-	2 (1,4)
LD	Rd, Y+	Charge Indirect Post Incrément	$Rd = (Y)$ et $Y = Y + 1$	-	2 (1,4)
LD	Rd, -Y	Charge Indirect Pré Décrément	$Y = Y - 1$ et $Rd = (Y)$	-	2 (1,4)
LDD	Rd, Y+q	Charge Indirect avec Déplacement	$Rd = (Y + q)$	-	2 (1,4)
LD	Rd, Z	Charge Indirect	$Rd = (Z)$	-	2 (2,4)
LD	Rd, Z+	Charge Indirect Post Incrément	$Rd = (Z)$ et $Z = Z + 1$	-	2 (2,4)
LD	Rd, -Z	Charge Indirect Pré Décrément	$Z = Z - 1$ et $Rd = (Z)$	-	2 (2,4)
LDD	Rd, Z+q	Charge Indirect avec Déplacement	$Rd = (Z + q)$	-	2 (1,4)
STS	k, Rr	Stock Direct en Mémoire	$(k) = Rr$	-	2 (1,4)
ST	X, Rr	Stock Indirect	$(X) = Rr$	-	2 (2,4)
ST	X+, Rr	Stock Indirect Post Incrément	$(X) = Rr$ et $X = X + 1$	-	2 (2,4)
ST	-X, Rr	Stock Indirect Pré Décrément	$X = X - 1$ et $(X) = Rr$	-	2 (2,4)
ST	Y, Rr	Stock Indirect	$(Y) = Rr$	-	2 (2,4)
ST	Y+, Rr	Stock Indirect Post Incrément	$(Y) = Rr$ et $Y = Y + 1$	-	2 (2,4)
ST	-Y, Rr	Stock Indirect Pré Décrément	$Y = Y - 1$ et $(Y) = Rr$	-	2 (2,4)
STD	Y+q,Rr	Stock Indirect avec Déplacement	$(Y + q) = Rr$	-	2 (1,4)
ST	Z, Rr	Stock Indirect	$(Z) = Rr$	-	2 (2,4)
ST	Z+, Rr	Stock Indirect et Incrément postal	$(Z) = Rr$ et $Z = Z + 1$	-	2 (2,4)
ST	-Z, Rr	Stock Indirect et Pré décroissance	$Z = Z - 1$ et $(Z) = Rr$	-	2 (2,4)
STD	Z+q,Rr	Stock Indirect avec Déplacement	$(Z + q) = Rr$	-	2 (1,4)
LPM		Mémoire Programme de Charge	$R0 = (Z)$	-	3 (3)
LPM	Rd, Z	Mémoire Programme de Charge	$Rd = (Z)$	-	3 (3)
LPM	Rd, Z+	Mémoire Programme de Charge Post Incrément	$Rd = (Z)$ et $Z = Z + 1$	-	3 (3)
ELPM	Extended	Mémoire Programme de Charge	$R0 = (RAMPZ:Z)$	-	3 (1)
ELPM	Rd, Z	Mémoire Programme de Charge Etendue	$Rd = (RAMPZ:Z)$	-	3 (1)
ELPM	Rd, Z+	Mémoire Programme de Charge Etendue Post Incrément	$Rd=(RAMPZ:Z)$ et $Z=Z + 1$	-	3 (1)
SPM		Stock Mémoire Programme	$(Z) = R1:R0$	-	- (1)
IN	Rd, A	Entrée d'un Port	$Rd = I/O(A)$	-	1
OUT	A, Rr	Sortie sur Port	$I/O(A) = Rr$	-	1
PUSH	Rr	Entrée du Registre de Pile	$STACK = Rr$	-	2 (1)
POP	Rd	Sortie du Registre de Pile	$Rd = STACK$	-	2 (1)

Code	Opér	Description	Opération	Drapeau	Cycle
<b>Instructions de Bit et Bit test</b>					
LSL	Rd	Décalage Logique à Gauche	$C=Rd(7)$ , $Rd(n+1)=Rd(n)$ , $Rd(0)=0$	Z C N V H	1
LSR	Rd	Décalage Logique à Droite	$C=Rd(0)$ , $Rd(n)=Rd(n+1)$ , $Rd(7)=0$	Z C N V	1
ROL	Rd	Rotation à Gauche	$Rd(0)=C$ , $Rd(n+1)=Rd(n)$ , $C=Rd(7)$	Z C N V H	1
ROR	Rd	Rotation à Droite	$Rd(7)=C$ , $Rd(n)=Rd(n+1)$ , $C=Rd(0)$	Z C N V	1
ASR	Rd	Décalage Arithmétique à Droite	$Rd(n) = Rd(n+1)$ , $n=0:6$	Z C N V	1
SWAP	Rd	Échange demi partie Réciproque	$Rd(3:0)=Rd(7:4)$ et $Rd(7:4)=Rd(0:3)$	-	1
BSET	s	Drapeau SREG(s) à 1	$SREG(s) = 1$	SREG(s)	1
BCLR	s	Drapeau SREG(s) à 0	$SREG(s) = 0$	SREG(s)	1
SBI	A, b	Registre d'entrée-sortie à 1	$I/O(A, b) = 1$	-	2
CBI	A, b	Registre d'entrée-sortie à 0	$I/O(A, b) = 0$	-	2
BST	Rr, b	Drapeau T = position b du Registre	$T = Rr(b)$	T	1
BLD	Rd, b	Registre position b = Drapeau T	$Rd(b) = T$	-	1
SEC		Drapeau Retenue à 1	$C = 1$	C	1
CLC		Drapeau Retenue à 0	$C = 0$	C	1
SEN		Drapeau Négatif à 1	$N = 1$	N	1
CLN		Drapeau Négatif à 0	$N = 0$	N	1
SEZ		Drapeau Zéro à 1	$Z = 1$	Z	1
CLZ		Drapeau Zéro à 0	$Z = 0$	Z	1
SEI		Drapeau d'Interruption Permit	$I = 1$	I	1
CLI		Drapeau d'Interruption Arrêté	$I = 0$	I	1
SES		Drapeau Signé à 1	$S = 1$	S	1
CLS		Drapeau Signé à 0	$S = 0$	S	1
SEV		Drapeau Débordement à 1	$V = 1$	V	1
CLV		Drapeau Débordement à 0	$V = 0$	V	1
SET		Drapeau Transfert à 1	$T = 1$	T	1
CLT		Purifiez Transfert à 0	$T = 0$	T	1
SEH		Drapeau Moitié de Retenue H à 1	$H = 1$	H	1
CLH		Drapeau Moitié de Retenue H à 0	$H = 0$	H	1

Code	Opérant	Description	Drapeau	Cycle
<b>Instruction de Contrôle MCU</b>				
BREAK		Voir la description de l'instruction de <b>BREAK</b>	-	1 (1)
NOP		Pas d'Opération (Attente d'un cycle)	-	1
SLEEP		Voir la description de l'instruction de <b>SLEEP</b>	-	1
WDR		Voir le chapitre sur le Watchdog ou l'instruction <b>WDR</b>	-	1

**Notes** 1 Cette instruction n'est pas disponible dans tous les modèles.

2. L'ensemble des variantes des instructions n'est pas disponible dans tous les modèles.

3. L'ensemble des variantes de l'instruction **LPM** n'est pas disponible dans tous les modèles.

4. Le cycle d'horloge pour les accès mémoire de Données n'est pas valable pour des accès via l'interface externe de **RAM**. Pour **LD**, **ST**, **LDS**, **STS**, **PUSH**, **POP**, ajouter un cycle de plus pour chaque état d'attend. Pour **CALL**, **ICALL**, **EICALL**, **RCALL**, **RET**, **RETI** avec **PC** à 16 bits, l'ajout de trois

cycles plus deux cycles pour chaque état d'attend. Pour **CALL**, **ICALL**, **EICALL**, **RCALL**, **RET**, **RETI** avec le **PC** à 22 bits, l'ajoute de cinq cycles plus trois cycles pour chaque état d'attend.

## Légende du jeu d'instruction

### Status Register (SREG)

**C** : Retenue,  
**Z** : Zéro,  
**N** : Négatif (bit 7 à 1),  
**V** : Débordement,  
**S** :  $\overline{N \wedge V}$ , pour les opérations signés,  
**H** : Moitié de Retenue (**DCB**),  
**T** : Bit de Transfert employée par **BLD** et **BST**,  
**I** : Interruption Permise ou Arrêté.

### Registres et Opérandes

**Rd** : Registre de Destination (ou de source) écrit,  
**Rr** : Registre source lue,  
**R** : Le résultat après l'instruction est exécuté,  
**K** : Données constantes,  
**k** : Adresse constante,  
**b** : Particule dans le Fichier de Registre ou Registre d'entrée-sortie (à 3 bits),  
**s** : Particule dans le Registre de Statut (à 3 bits),  
**X, Y, Z** : Registre d'Adresse Indirect ( $X=R27:R26$ ,  $Y=R29:R28$  et  $Z=R31:R30$ ),  
**A** : Adresse d'emplacement d'entrée-sortie,  
**q** : Déplacement pour adressage direct (à 6 bits).

### RAMPX, RAMPY, RAMPZ, RAMPD

Les registres sont enchaînés avec **X-**, **Y-** et **Z-** permettant l'adressage indirect de l'espace de données du **MCU** en débordant l'espace de données de 64Ko.

Registre enchaîné **Z-** permettant l'adressage direct de l'espace de données entier du **MCU** en débordant l'espace de données de 64 **Ko**.

### EIND

Registre enchaîné avec le mot d'instruction permettant un appel indirect sur l'espace programme entier du **MCU** en débordant l'espace programme de 64 **Ko**.

### STACK

La pile pour le retour d'adresse des registres poussés dans **SP**, Indicateur de Pile pour empiler.

### DRAPEAU

= : Drapeau affecté par instruction  
**0** : Drapeau purifié par instruction  
**1** : Drapeau mis par instruction  
- : Drapeau non affecté par instruction