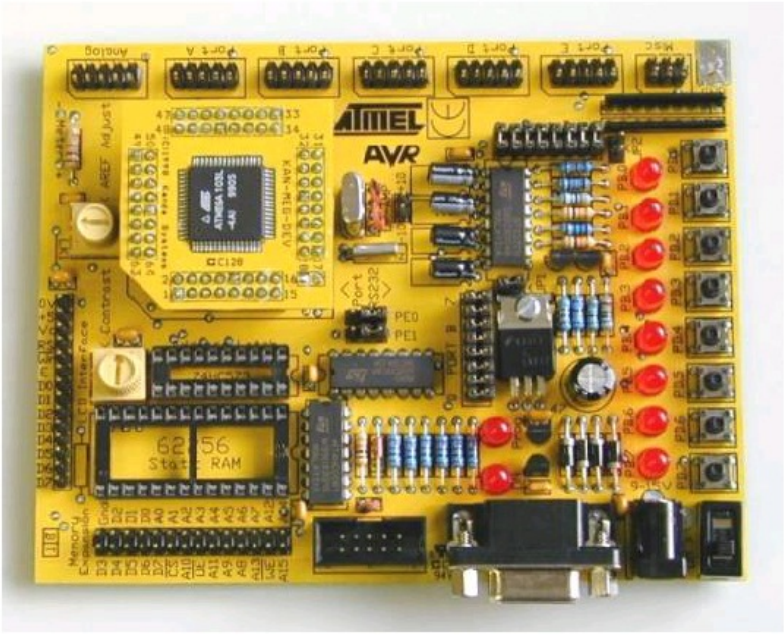
SÉCTION DE MICROTECHNIQUE

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Cahier des charges

PROJET MORSE

_ _ _ _ _ * _ * * * * *



Auteurs	Tristan Besson. Dylan Bourgeois
Numéro de groupe	M3
Assistant responsable	
Enseignant	M. Schmid
Affiliation	MT
Semestre	BA3
Cours	Microcontroleurs
Date	28/04/2014

1. Introduction

Le langage mis au point par Samuel Morse en Juin 1940 permit un essor des systèmes de communications. Il permit de relier plus rapidement que jamais les côtes Ouest et Est des États-Unis. Les lignes télégraphiques se retrouvèrent ainsi vite saturée face à une demande grandissante.
L'amélioration fulgurante des moyens de communications dans les dernières décennies réduisit le Morse a un langage de secours en cas de force majeur. Pourtant, peu de gens sont désormais à même de pouvoir le décoder.
Notre projet repose donc sur une idée simple: pouvoir traduire un mot codé en morse.

2. Cahier des charges

Pour notre projet, nous avons choisi de traduire le morse « sonore » et par conséquent, nous utilisons comme périphérique le microphone MP33125, couplé avec l'amplificateur NE5534AN. Le signal émis par le microphone varie autour de 2,5 V entre +5 V et 0 V et arrive sur le deuxième pin du port analogique.
Nous avons décidé d'analyser des sons d'une fréquence de 440 Hz avec pour durée d'un dot « ° » de 131ms. Le choix de cette période est motivé par l'obtention d'une valeur suffisamment audible et en même temps s'accorder aux possibles overflow des timers à notre disposition.
Notre microcontrôleur devra ainsi être à même de pouvoir traduire un mot selon le chemin des données suivant:

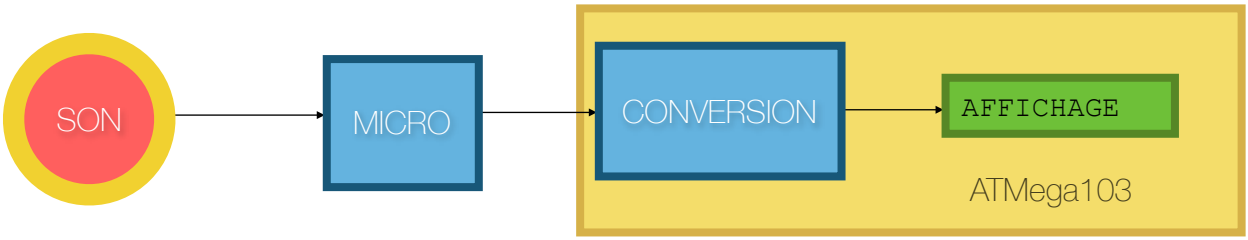


Figure 1: Chemin des données

3. Fonctionnement du programme

1. Détection de la présence d'un son

Le signal arrivant sur le port analogique du microcontrôleur est une simple fonction sinusoïdale. Nous commençons directement par ignorer les valeurs qui se situent sous un certain seuil afin de limiter les fausses mesures occasionnées par le bruit ambiant de la salle. Une détection triviale du son consisterait à prendre un ensemble de valeur et à en faire la moyenne.

Cette méthode ne peut être utilisée dans le cas actuel. En effet, nous aurions la même moyenne que lors de l'absence de son.

Nous avons donc décidé d'utiliser une méthode qui compte les pics de tension sur la ligne. Les mesures étant relativement proche, le risque de compter plusieurs fois un même pic est important. Ainsi, le flag "**c2**" est activé pour signaler que nous sommes toujours sur un même pic et donc nous n'incrémentons pas le compteur de pics "**c0**". Le flag "**c2**" est remis à 0 dès que le signal repasse en dessous du seuil.

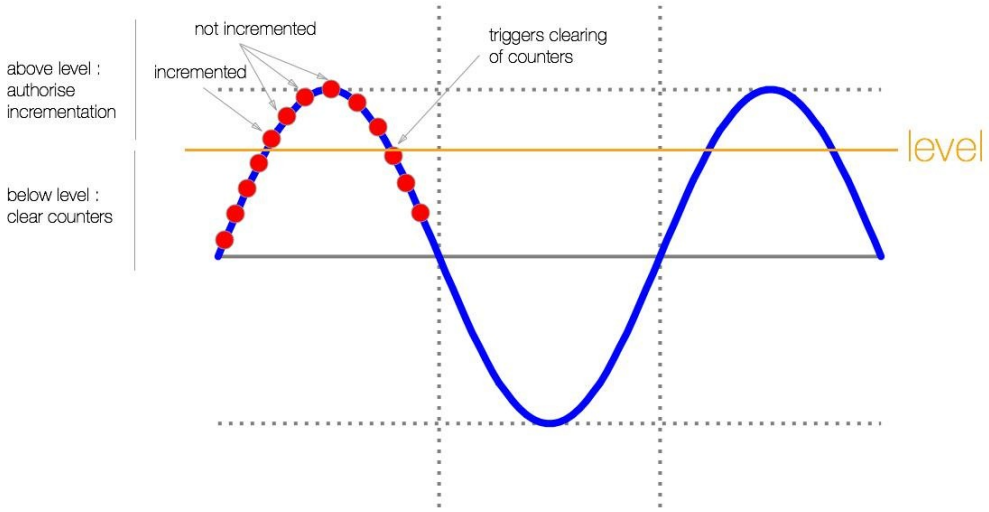


Figure 2: Système de détection d'un son

Afin de placer une échelle de temps sur notre décodage, nous avons opté pour le timer1 avec un prescaler à 2 afin d'avoir un overflow au bout de 131ms (la durée exacte d'un " ° "). Lors de l'overflow, la routine d'interruption **analyse** est appelée. Elle analyse le nombre de pics obtenus durant la plage de temps et la compare à la valeur de 64, valeur à partir de laquelle nous estimons qu'un son a réellement été détecté. Si on estime qu'un son a été detecté, le compteur de vide **c1** est remis à 0 et le compteur de pulses **d2** est incrementé. En l'absence de son, la routine **shortlong** est appelée, puis le compteur de pulses est remis à 0 et le compteur de vide incrémenté.

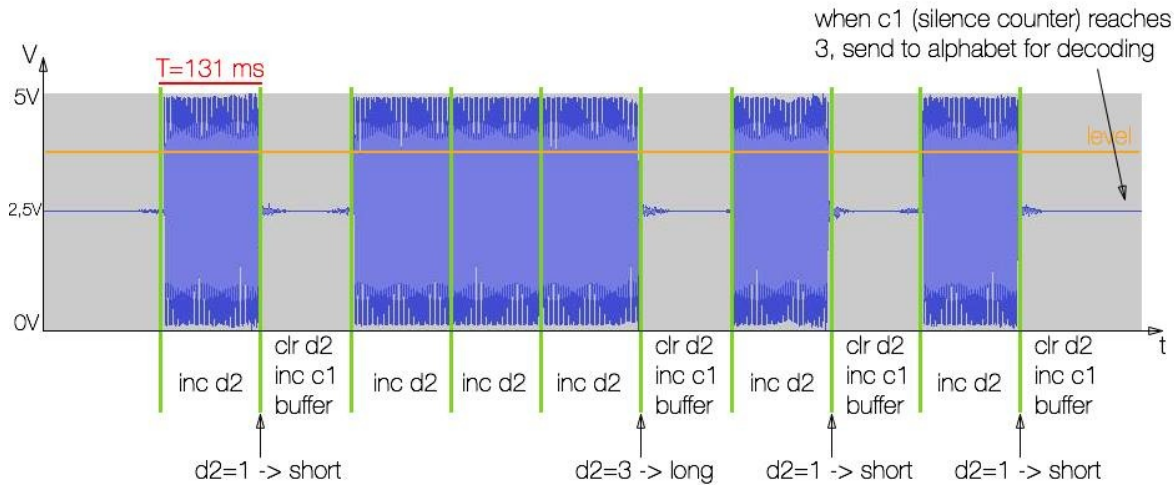


Figure 3: Analyse d'un enchainement de pulses

2. Différenciation des sons courts et longs

Dans un premier temps, nous devons savoir si nous sommes à la fin d'une lettre. Pour cela nous testons la valeur du compteur de vide (**c1**) : s'il est à trois, nous pouvons envoyer le buffer à décoder par la routine *alphabet*.

Si tel n'est pas le cas, nous devons déterminer si le son retenu est court ou long. Là encore nous testons un compteur qui a été incrémenté selon l'impulsion entrante. Nous recevons là le nombre de pulses consécutifs où un son a été détecté. Ainsi s'il est à trois, nous pouvons faire entrer un son long dans le buffer, s'il est à un, nous pouvons entrer un son court.

3. Buffer

Nous ne pouvons pas envoyer à décoder à chaque fois qu'un signal (court ou long) arrive, car chaque lettre est encodée en morse selon un enchainement unique de ces sons longs et courts. Il faut donc avoir un registre qui servirait de buffer, c'est-à-dire qui stocke au fur et à mesure les valeurs des signaux entrants, pour ensuite envoyer cet enchaînement lorsque le signal encodant la lettre est terminé.

Nous devons d'abord nous mettre d'accord sur une symbolique pour représenter ces signaux. Nous avons choisi d'utiliser 2 bits pour encoder les 3 possibilités.

Encodage des signaux

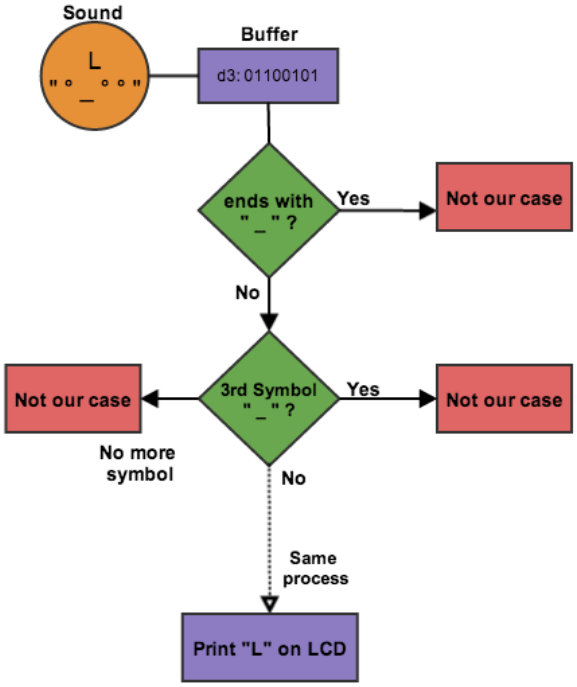
signal court	01
signal long	10
signal absent	00
don't care	11

A chaque fois qu'une valeur de signal (court ou long) arrive, nous vérifions sa valeur afin de la classifier correctement. Une fois que nous avons déterminé si le signal est court ou long, nous pouvons décaler le registre de buffer de 2 vers la gauche. En libérant les deux LSB, nous pouvons alors insérer par simple addition la valeur arrivante.

Comme dit précédemment, lorsque trois vides sont détécés à la suite, nous pouvons envoyer le contenu du buffer à la routine de décodage de l'alphabet, qui assigne un caractère à l'enchainement de sons courts et longs. Enfin le buffer est vidé pour accueillir la prochaine lettre.

4. Buffer et correspondance dans l'alphabet

En partant du principe que toutes les lettres n'ont pas le même nombre de symboles pour être encodées, nous ne pouvons pas lire le registre de gauche à droite, mais plutôt de droite à gauche. On utilise la technique du "divide and conquer" pour séparer le problème à chaque branche.



La routine alphabet fonctionne comme un arbre. À chaque ramification, la même technique est utilisé: est-ce que le symbole suivant existe? est-ce un dot ou un dash?
Ainsi, la routine avance dans sa recherche de la lettre jusqu'à arriver aux routines print_"letter" qui se chargent d'écrire le symbole trouvé sur l'écran.

Figure 4: Recherche de la lettre L

5. Schémas Récapitulatifs

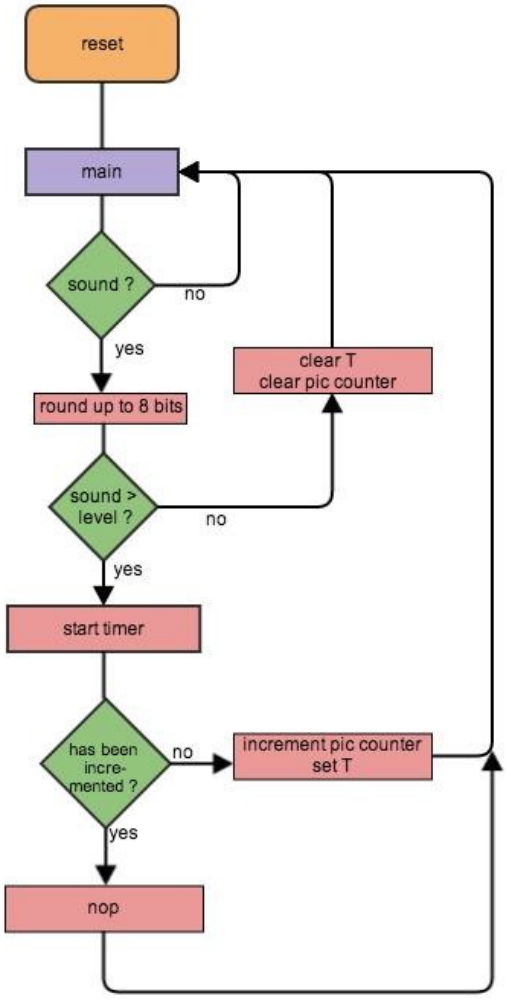


Figure 5 : Fonctionnement général du programme autour du main

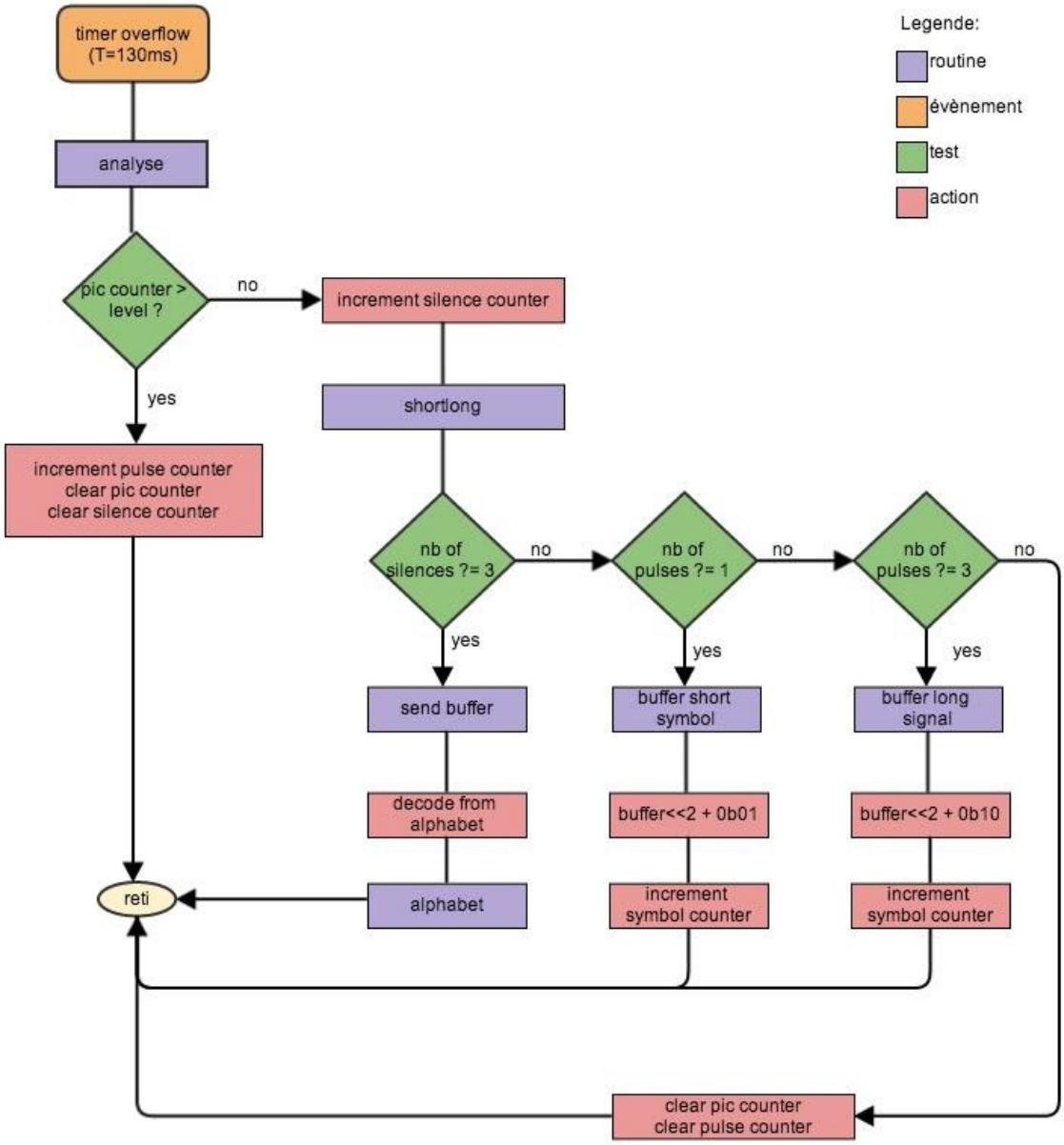


Figure 6: Fonctionnement général du programme à partir d'un overflow du timer

4. Conclusion

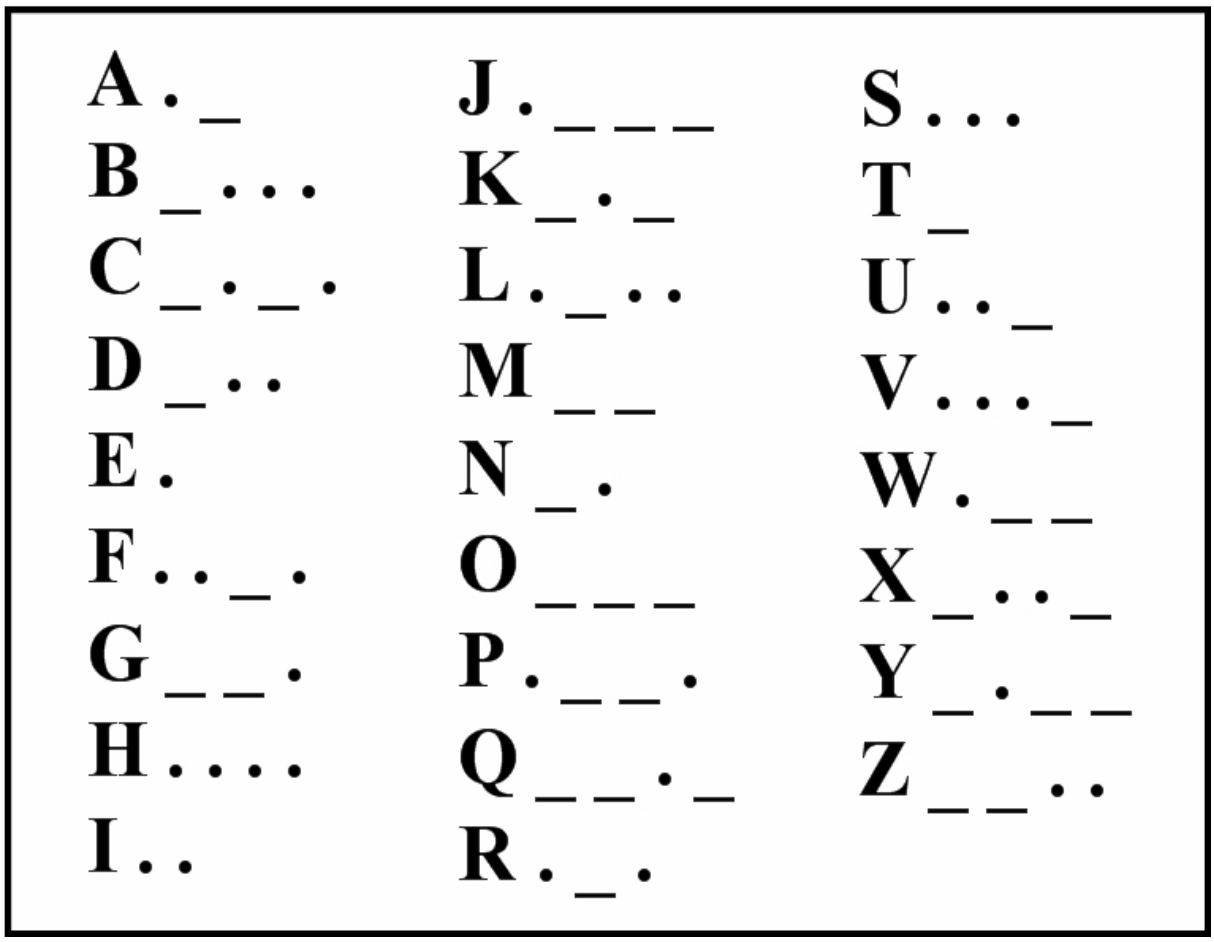
Ce projet nous a permis de mettre en pratique les connaissances assimilées durant le semestre. Malgré quelques incertitudes par moment, notre microcontrôleur fonctionne et arrive à traduire des mots comme "WORLD".

Néanmoins, le projet pourrait être nettement améliorer si l'on pouvait adapter le microcontrôleur à n'importe quelle période de son pour un "Dot", aussi pour n'importe quelle fréquence. Ceci pourrait être possible en plaçant comme premier son du message une sorte de calibrage qui permette au microcontrôleur d'adapter son nombre de pics nécessaires à la détection d'un son (fréquence) et adapter le prescaler du timer1 (période).

Figure : Routines du programme

routine	in	out	description
analyse	c0	d2	interruption routine. Detects if sound was present during period.
reset	-	-	Reset routine. Set timers, analog digital, define ports, init LCD and registers.
clr_init	-	-	clears all registers used in program
clr_timer	-	-	resets timer counter register TCNT1H/L
main	d0,d1	-	main routine. Counts the number of pics of sound continuously.
sound	c2,c3	c2,c0	called if sound level in main is superior to level. Increments pic counter if it wasn't already in this cycle.
nosound	c2	-	called if sound level is inferior to level. clears c2
shortlong	c1,d2	-	determines if a sound is short or long
buffershort	d3	b3	buffer a short value. Increments signal counter
bufferlong	d3	b3	buffer a long value
sendbuffer	d3	-	sends the buffer to be decoded
alphabet	d3	-	divides the decoding into 2 parts : letter starts with short or long.
alphabet_dot	d3	-	decodes the value of the signal to a character
alphabet_dash	d3	-	decodes the value of the signal to a character

5. Annexe : L'alphabet morse



name	value	usage
MORSE_PORT	2	define analog port to be used
LEVEL_NB_PICS	64	define level fro number of pics
LEVEL_SND	63	define sound strength level
AD_PS	5	define AD prescaler (f=125kHz)
TIMER_PS	2	define prescaler (T=131ms)
SIGNED	127	define constant to bring back counter to signed
SHORT_SYMB	0b01	define buffer symbolic for short signal
LONG_SYMB	0b10	define buffer symbolic for short signal

Figure: Macros du projet

register	denomination	utility
r8	c0	pic counter
r9	c1	silence counter
r10	c2	did counter increment on this pulse
r11	c3	did timer start
r12	d0	ACDH
r13	d1	ADCL
r14	d2	pulse counter
r15	d3	buffer register
r16	w	working register
r17	_w	working register
r18	a0	-
r19	a1	-
r20	a2	-
r21	a3	-
r22	b0	sound level value
r23	b1	sound level compare register
r24	b2	-
r25	b3	character counter

Figure: Utilisation des registres