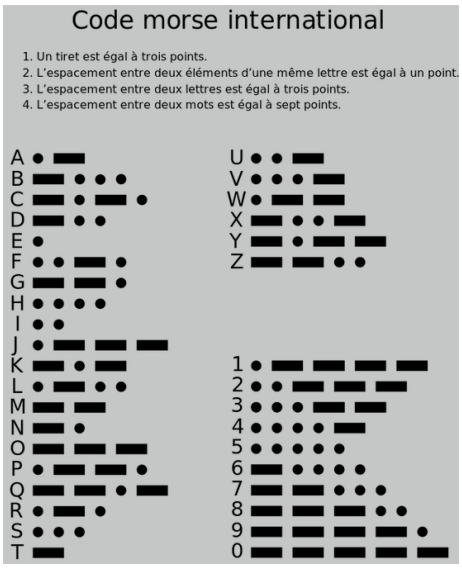
TP TEST IE
Conception d'un emetteur Recepteur Morse

Lire l'intégralité du sujet avant de commencer le TP, sa difficulté est progressive. *Tous les Comptes Rendus personels et les sujets des précédents TP sont autorisés*

1 Introduction

L'objectif de ce TP est de réaliser un encodeur/décodeur de code MORSE, en faisant appel à vos connaissances sur les protocoles de communication UART, l'utilisation du GPIO et des timers.

> *Le code Morse international, ou l'alphabet Morse international, est un code permettant de transmettre un texte à l'aide de séries d'impulsions courtes et longues, qu'elles soient produites par des signes, une lumière, un son ou un geste.*



1.1 Convention et cadence

- Le rythme élementaire est donné par la durée du point “ti”,
- un “taah” est conventionnellement 3 fois plus long qu'un “ti” Il se note par un trait horizontal “–”
- L'espacement entre deux lettres à pour longueur un “taah”, il se note par un espace.
- L'espacement entre deux mots est de 7 ti (ou “deux taah” et un “ti”), il se note par une barre oblique,

```
-.-. --- -.. . / -- --- .-. ... .
« taahtitaahti taahtaahtaah taahtiti ti, taahtaah taahtaahtaah titaahti tititi ti
"code morse"
```

A partir de ces informations, on codera le “ti” par `0001` et “taah” par `0111` , en lisant l'etat de droite à gauche. Ainsi, le message “ALLO”: sera reçu (en lisant de gauche à droite!) comme `1000 1110` (A), `1000 1110 1000 1000` (L), `1000 1110 1000 1000` (L), `1110 1110 1110` (O)

1.2 Objectif de la séance

On se propose donc de créer un transcodeur morse constitué:

- Un **encodeur**: En utilisant l'hyperterminal UART, l'utilisateur entre un message à décoder, et le code est retransmis grâce aux LED du micro controleur. On utilisera à cette fin la bibliothèque `morse.h`
- Un **décodeur**: l'utilisateur entre lettre après lettre (en utilisant un bouton poussoir en rythme) le code pour faire un mot. Ce dernier est affiché via l'écran LCD puis envoyer par UART.
- L'utilisation des bibliothèques `GPIO.c`, `UART.c`, `TIMER.c` est autorisée.

1.3 Rappel: Opération BitShift

L'operation de bit-shift (vers la droite `>>` ou vers la gauche `<<`) correspond à un décallage des bits vers la droite ou la gauche. Cela est équivalent à une multiplication par 2^{n} (`<<n`) ou une division (`>>n`) par 2^{n}. Exemple

- $54 \ll 3 = \overline{0011\ 0110} \ll 3 = \overline{1\ 1011\ 0000} = 432$. NB: Si 54 était un `char`, on aurait seulement $\overline{1011\ 0000} = 176$)

- $54 \gg 3 = \overline{0011\ 0110} \gg 3 = \overline{0000\ 0110} = 6$.

2 Préparation

1. Création du projet
   - Créer un nouveau projet Keil (dans `Documents/`, la compilation sera *a priori* plus rapide).
   - Copier les bibliothèques usuelles (disponible dans `Communs/IENA/S2/`).
   - Créer un nouveau fichier `morse.h` et `morse.c` y ajouter le code fournis en annexe. Les bibliothèque sont également accessible en ligne
   - Créer un nouveau fichier `main.c`, et y ajouter le code fournis en annexe.
2. Lire le code de `morse.c`.
   - A quelle lettre correspond le nombre 373 dans le tableau `MORSE_CODE`? Expliquer le codage utilisé.
   - Que fait la fonction `char2morse(char letter)` ?
3. Pour afficher ce qui se passe on utilise les LED présentent sur la carte, ici le pin `P2.6`. Écrire la fonction `void morseLED(int status)` qui commande cette led (allumée si `status > 1`, éteinte sinon).
4. Demander à l'encadrant un câble série pour la communication UART avec le PC.

3 Encodeur Morse

5. Configurer l'hyper-terminal sur le PC pour communiquer avec le microcontrolleur. (on choisira le protocole UART 8 bits de signal, 1 parité 2 stops).
   - Quel est la configuration UART utilisé par la fonction `void InitUART1(char parite)`? Donner le nombre de bits de données, de stop et le baud-rate utilisé. On utilisera une parité paire.
   - Quel est le codage standard utilisé pour transmettre des caractères sur cette liaison UART ?
6. A l'aide de l'hyper-terminal envoyer une lettre par liaison UART depuis le PC et faites afficher son code morse a l'aide de la fonction `void char2morseblink(char letter)`. Cette fonction fait appel à `morseLED(int status)`.
7. Pour mieux visualiser le code morse associé à une lettre, on utilise les 8 LED de la

carte, à la manière d'un chenillard.Le comportement attendu est le suivant pour l'envoi de "ALLO": (Avec LED1=la led la plus à gauche, LED8 la plus à droite)

- Quel sera l'état des Leds à t+9 et t+12 ?
- Compléter la fonction `void char2morsetrail(char letter)`. *Astuce:* On pourra utiliser `void Ecriture_GPIO(unsigned char Valeur)` pour commander les 8 leds simultanément.

Table 1: Chronogramme d'affichage du message 'ALLO' en morse sur la serie de LED.

LED	1	2	3	4	5	6	7	8	Lecture
t+1	1	0	0	0	0	0	0	0	
t+2	0	1	0	0	0	0	0	0	
t+3	0	0	0	1	0	0	0	0	'.'0000
t+4	1	0	0	0	1	0	0	0	1'.'000
t+5	1	1	0	0	0	1	0	0	11'.'00
t+6	1	1	1	0	0	0	1	0	111'.'0
t+7	0	1	1	1	0	0	0	1	'-.'→'A'
t+8	1	0	1	1	1	0	0	0	1'-'000
t+9									
...									
t+12									

8. Réaliser et exécuter un programme qui:
   - Écoute le canal de transmission UART1 et récupère les caractère ASCII envoyé depuis le PC.
   - Traduit le caractère en morse.
   - Affiche son code grâce aux LED (au choix en clignotant ou en utilisant le chenillard).

4 Décodeur Morse

9. Pour faciliter l'entrée d'un code morse, on se propose de se doter d'un battement de référence, en faisant clignoter une LED à la fréquence de 2Hz.
   - Configurer le timer TIM0 (`prescaler` et `N` pour avoir un signal carré de 2Hz. (On pourra adapter la fonction `void signalCarre()` du TP5)
   - Faire clignoter la LED `P2.5` en utilisant le timer (pour s'affranchir des imprécisions du processeur, on utilisera `LPC_TIM0->TC >= N1` avec `N1<N`)

- A quel mécanisme vu en cours pourrait-on faire appel pour garantir l'instant de lecture des boutons ?

10. Pour décoder du morse facilement a la machine, on va utilise le joystick avec ces pins `P0.18` et `P0.16`. Appuyer sur `P0.18` correspond a un "ti" (point), `P0.16` à un "taah".
    - Ecrire un programme qui lit les entrées du joystick en rythme avec la led (ie quand la led s'allume) et encode le caractère suivant le codage de `MORSE_CODE`.
    - Décoder cette lettre et l'envoyer par UART
11. On souhaite maintenant développer les fonctionnalités du décodeur, implémenter par exemples les fonctions suivantes:
    - Affichage lettres après lettre sur l'écran LCD
    - Décodage à la vollée du code morse (par exemple "ti ti ti ti" va afficher successivement "?E, ?I,? S, H")

-. . / .–. . .-. -.. . –.. / .–. .- . . . / -.. . / - . – .–. . . . / .- / -.. . -.-. . . . . ..
..-. ..-. .-. . .-. / -.-. . / – . . . .- –. . .-.-.-

5 Annexes

```c
// main.c
#include <LPC17xx.h>
#include "GPIO.h"
#include "UART.h"

int main(void){

    return 0;
}
```

Listing 1: main.c

```c
// morse.h

void dotDelay();
int char2morse(char letter);
void char2morseblink(char letter);
void char2morsetrail(char letter);

char morse2char(int morse);
void str2morse(char* msg, int size);
int btn2morse(void);

void morseLED(int status); // A écrire par l'étudiant (partie 2)
void errorLED(); // A écrire par l'étudiant (partie 2)
void timerLED(int status); // A écrire par l'étudiant (partie 4)
```

Listing 2: morse.h

```c
// morse.c
#include "morse.h"
// A to Z
int MORSE_CODE[] = {29, 343,1495,87, 1,373,375,213,5,7645,471,349,119,23,1911,
1501,7543,93,53,7,117,469,477,1879,7639,1399};

// À compléter
void morseLED(int status){
if(status > 0){
    // allumer la led P2.6
}
else{
    // eteindre la led P2.6
}
}

// À compléter
void timerLED(int status){
if(status > 0){
    // allumer la led P2.5
}
else{
    // eteindre la led P2.5
}
}

void dotDelay(){
        int i = 0;
        for(i=0; i< 8500000; i++); // 0.5s delay
}

int char2morse(char letter){
        int code;
        if (letter >= 'a' && letter <='z'){code = MORSE_CODE[letter - 'a';}
        else if(letter >= 'A' && letter <= 'Z'){code = MORSE_CODE[letter - 'A';}
        else if(letter == ' '){code=0;}
        return code;
}

void char2morseblink(char letter){
        int code;
        int i = 0;
        code = char2morse(letter);
```

```
        while(1){
                dotDelay();
                if(!code){
                        morseLED(0);
                        break; // rien a afficher, on quitte la boucle.
                }
                else{
    // A compléter
                        //morseLED(<dernier_bit de code>);
                }
   // A compléter
  // Decaler les bits de 'code' d'un cran vers la droite.
}
}

void char2morsetrail(char letter){
        int code;
        int i = 0;
        code = char2morse(letter);
        if(!code){return;} // ne rien faire si pas de code.

        code = code << 8; // On se créer un espace de 8 bit devant
        while(code){
                dotDelay();
  // A Compléter
  // Afficher les 8 bits de poids faibles de code.
  // Decaler les bits de 'code' d'un cran vers la droite (bit shift)
        }
        // A Compléter
 // Eteindre toutes les LEDS.
}


void str2morse(char* msg, int size){
int i = 0;
for(i=0; i < size; i++){
        char2morsetrail(msg[i]);
        dotDelay();
}
}

char morse2char(int morse){
char i=0;
```

```
for(i=0; i < 26; i++){
    if(morse == MORSE_CODE[i]) return i + 'A';
}
return '?';
}
```