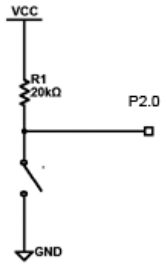
C5-160551-INFO
Objet Connectés : Programmation microcontrôleur
TD révision

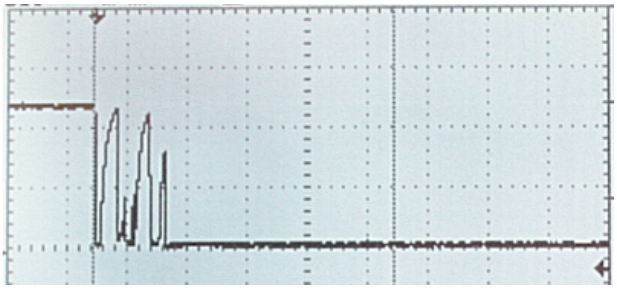
©B. Besserer

année universitaire 2021-2022

1 REVISION

1. Examinez les figures ci-dessous :





a) Quelle est la fonction de la résistance R1 ?

- [ ] A éviter un court-circuit quand l'interrupteur est fermé.
- [ ] A assurer un niveau logique 1 sur la broche microcontrôleur (noté P2.0, et configurée en entrée) lorsque l'interrupteur est ouvert.
- [ ] A assurer un niveau logique 0 sur la broche microcontrôleur (noté P2.0, et configurée en entrée) lorsque l'interrupteur est ouvert.
- [ ] A assurer un niveau logique 1 sur la broche microcontrôleur (noté P2.0, et configurée en entrée) lorsque l'interrupteur est fermé.
- [ ] A éviter les courts-circuits lorsqu'on configure la broche en sortie.

b) Des résistances assurant la même fonction sont maintenant intégrés dans les microcontrôleurs modernes,,dont l'ESP32. Quel est la ligne de code pour une configuration correcte, en entrée, de manière a ce que la résistance R1 devienne superflue ?

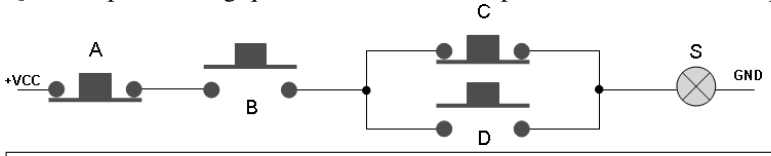
c) Que représente le chronogramme (à droite) relevé à l'oscilloscope sur la broche P2.0 ?

- [ ] Des parasites électromagnétiques captés sur P2.0 car le branchement de l'interrupteur tel que montré sur le schéma est incorrect
- [ ] Des rebonds provoqué par la fermeture de l'interrupteur (rebonds mécaniques de la lame qui assure le contact)
- [ ] Un phénomène qui arrive lorsqu'on a initialisé le GPIO comme sortie et que l'on s'en sert comme entrée

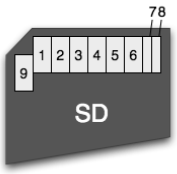
d) Quelle est approximativement la durée du phénomène que l'on voit sur le tracé de l'oscilloscope ?

- [ ] Quelques nanosecondes
- [ ] Quelques microsecondes
- [ ] Quelques millisecondes
- [ ] Quelques secondes

2. Quelle expression logique combinatoire correspond à ce schéma électrique (expression logique de S) ?



3. Dans une application, des résultats doivent être consignés par l'ESP32 dans une mémoire externe pour relecture ultérieure, on choisi donc d'utiliser une carte SD. Ci-contre le brochage. Quel type de liaison permet l'interface avec une carte SD ?



Pin	SPI
1	CS
2	DI
3	VSS1
4	VDD
5	SCLK
6	VSS2
7	DO
8	X
9	X

- ☐ Une liaison paralléle
- ☐ Une liaison série synchrone SPI
- ☐ Une liaison série asynchrone RS232
- ☐ Un bus série I^2C
- ☐ Un bus série CAN

4. Sur l'ESP32, pour les numéros de broches (GPIO) inférieurs à 32, le code qui correspond à digitalRead() peut se résumer au code suivant :

```
extern int IRAM_ATTR digitalRead(uint8_t pin)
{
    return (GPIO.in >> pin) & 0x1;
}
```

a) Décrivez brièvement ce que fait cette ligne de code ?

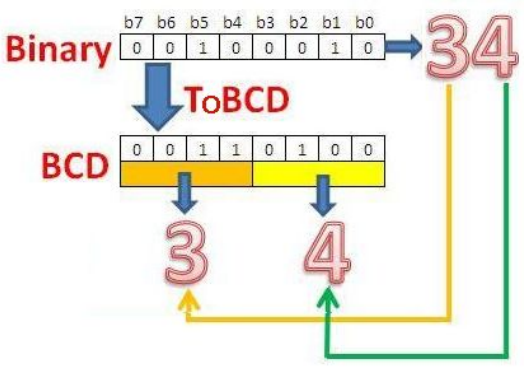
b) Pourquoi `IRAM_ATTR` ?

c) Ici, le modificateur "extern" est utilisé dans le même but que "volatile", on aurait pu ecrire `volatile int IRAM_ATTR digitalRead(uint8_t pin)`. Quel est le rôle de ce modificateur ?

2 REVISION Programmation microcontrôleur

Le format BCD (Binaire Coded Decimal) est souvent utilisé quand on travaille avec des microcontrôleurs notamment lorsqu'on souhaite afficher des nombres sur des afficheurs 7 segments : il faut alors décomposer le nombre en unités, dizaines, centaines, milliers, etc... et envoyer la valeur des unités vers l'afficheur des unités, les dizaines vers l'afficheur des dizaines, etc...

Alors plutôt que de consacrer une variable pour les unités (valeurs de 0 à 9) et une variable pour les dizaines (valeurs de 0 à 9), le format BCD (Binary Coded Decimal) a été souvent utilisé, format dans lequel un **octet** sert à représenter un nombre de 00 à 99, sachant qu'un groupe de 4 bits (*nibble*) représente un chiffre de 0 à 9.
Mais attention, lorsqu'un octet contient un nombre au format BCD, on ne peux plus l'utiliser par exemple pour des opérations arithmétiques car l'ALU du microcontrôleur ignore tout de ce format (précisons que la représentation BCD des nombres n'est pas non plus un type de donnée existant en langage 'C'). Il faut donc des fonctions de conversion :
`byte BinaryToBCD(byte val)`
qui converti une valeur binaire dans sa représentation BCD.



Dans l'exemple, la valeur à convertir serait 34, la représentation BCD de ce nombre consiste à placer le code binaire des dizaines (3) dans le *nibble* de poids fort et le code binaire des unités (4) dans le *nibble* de poids faible de l'octet renvoyé par la fonction de conversion.

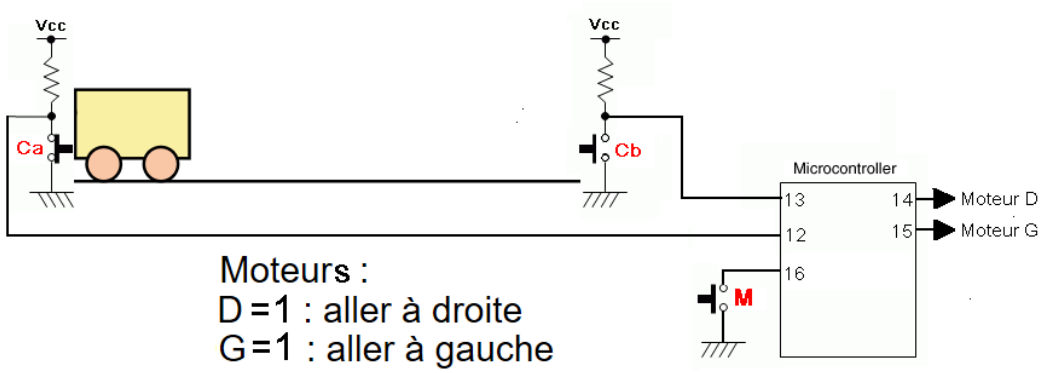
a) quel est le domaine de validité pour le paramètre `val` ?

b) Ecrivez la fonction `byte BinaryToBCD(byte val)` ... 4 à 5 lignes de code max...

c) Modifiez ou écrivez cette fonction pour une implémentation destinée à de "petits microcontrôleurs" sans utiliser de division, de multiplication ou d'opérateur modulo. (moins de 10 lignes)

3 REVISION Chariot bidirectionnel

Soit le dispositif suivant, qui sera contrôlé par un ESP32 qui sera programmé en utilisant le Framework Arduino. D et G sont des moteurs, commandés par les GPIO 14 et 15 respectivement. Ces moteurs fonctionnent lorsque le GPIO correspondant est au niveau HIGH, et chaque moteur entraine le chariot dans la direction indiquée (il faut éviter de faire fonctionner D et G en même temps). Ca et Cb sont des contacts de fin de course, reliés respectivement aux GPIO 17 et 18. M est un bouton poussoir pour la mise en marche du cycle, relié au GPIO16. On considère les contacts Ca, Cb et M sans rebonds.



Lorsque l'opérateur appuie sur M, le chariot doit se déplacer vers la droite jusqu'à actionner Cb, puis revenir vers la gauche jusqu'à toucher Ca et s'immobiliser en attente d'un nouveau cycle. On considère que l'opérateur relâche M avant que le chariot atteigne Cb, et ne fait pas d'autres actions sur M jusqu'à la fin du cycle.

Écrivez le code qui permette ce fonctionnement. Voir définitions ci-contre.
La vérification des état des entrées (Ca, Cb, M) se fera en mode pooling, pas de mise en veille ou d'interruption.
Selon votre style de codage, compter environ 5 à 7 lignes dans setup() et une dizaine de lignes dans loop()

```
#define Ca 17
#define Cb 18
#define D 14
#define G 15
#define M 16
void setup()
{
}
void loop()
{
}
```

En considérant le même système que précédemment (chariot bidirectionnel) et le même mode de fonctionnement, on souhaite maintenant pouvoir lancer le cycle avec un action sur une page HTML affiché sur le smartphone. Soit le code HTML suivant (qui donne l'affichage montré dans l'image). Ce code HTML est défini dans un fichier page.h en tant que raw string, nommé PAGE

```
<!doctype html>
<head>
  <title>Titre - pas visible sur smartphone</title>
</head>
<body>
<h1>Controle du chariot</h1>
</body>
<script>
var button = document.createElement("button");        // création d'une bouton
button.innerHTML = "Depart cycle";                    // texte du bouton
var body = document.getElementsByTagName("body")[0];

// Insertion dans le corps (body) de la page HTML
body.appendChild(button);

// Ajout d'un evenement au clic : navigation vers marche
button.addEventListener ("click", function() {
  window.location = "/marche";
});
</script>
</html>
```

Ci-dessous le squelette de code destiné à l'ESP32. Complétez pour obtenir le même fonctionnement avec une mise en marche via l'interface Web.

```
//on considère qu'il y les fichiers .h nécéssaires, y compris page.h
AsyncWebServer server(80);
const char* ssid = "OCPM";                            // nom du hotspot qui sera créé
void notFound(AsyncWebServerRequest *request) {       // callback pour navigation non conforme
  request->send(404, "text/plain", "Not found");
}

void setup() {
  Serial.begin(115200);
  ...                                                 // autres initialisations
  Serial.print("Setting AP (Access Point)...");
  WiFi.softAP(ssid);                                  // Creation d'un réseau ouvert, pas de mot de passe
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
                                                      // mise en place des callbacks
  server.onNotFound(notFound);
  ...                                                 // autres callbacks
  server.begin();
}
loop() {
   ...                                                // completer
}
```