

1 Logique combinatoire et logique séquentielle

1.1 Jeu logique

On conçoit la logique d'un jeu de déduction. Deux joueurs X et Y s'affrontent en choisissant un niveau logique 0 (ou *false* ou LOW) ou 1 (*true* ou HIGH) en présence d'un arbitre A qui fait également un choix. **Le but des joueurs est d'effectuer le même choix que l'arbitre.** Les résultats peuvent être les suivants :

- Le joueur X gagne et Y perd, on allume une LED rouge, que l'on nommera SX
- Le joueur Y gagne et X perd, on allume une LED verte, SY
- Match nul (X et Y gagnent ou bien X et Y perdent), on allume une LED jaune, SN.

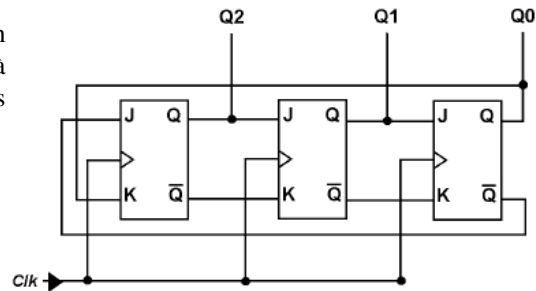
1. Définissez le système combinatoire : combien d'entrées ? combien de sorties (on souhaite visualiser les tours de jeu dont le résultat est nul) ? Puis donnez les équations des sorties (vous pouvez vous aider en traçant une table de vérité).
2. On souhaite implémenter ces règles dans un programme pour l'ESP32. Les valeurs X,Y et A sont lues sur les GPIOs 12,13 et 14 (boutons poussoir, et pour correspondre avec ce problème de logique combinatoire, ils sont câblés pour passer à l'état HIGH si appuyé, donc avec des résistances *pull-down*). Un autre poussoir (pareillement câblé) est relié au GPIO 15 : un appui sur celui-ci lance un compte à rebours de 5 secondes, faisant clignoter la LED_BUILDIN toute les secondes. Lorsque le compte à rebours atteint 0, les valeurs X,Y et A sont prises en compte.
 - Écrivez l'initialisation des broches à mettre dans la fonction `setup()`
 - Écrivez le code qui correspond au lancement du compte à rebours, à la prise en compte des valeurs à la fin du compte à rebours et les expressions logiques en langage C qui correspondent aux sorties.
3. Pour réaliser ce jeu sans microcontrôleur, il faut un circuit faisant office de timer et une entrée logique de plus, correspondant à la validation en fin de compte à rebours. On nomme cette entrée C, et C passe au niveau logique HIGH lorsque le compte à rebours expire. Par quelle fonction logique faut-il lier les sorties précédemment déterminées à C pour que les LED branchés sur les 3 sorties (LED qui indiquent le résultat) n'affiche l'information que lorsque C passe à HIGH ?

☐ fonction ET ☐ Fonction OU ☐ Fonction XOR (ou exclusif)
4. On souhaite maintenant figer le résultat : lorsque C passe à HIGH, les valeurs des sorties sont mémorisées et ne peuvent plus être changés, jusqu'à la fin du prochain compte à rebours (C repasse à LOW, puis de nouveau à HIGH).
 - Quel type de bascule synchrone convient pour figer cette information (seul un choix parmi ces 3 ne peut **pas** convenir)
 - ☐ Bascule JK ☐ Bascule D ☐ Bascule T
 - Quel signal d'entrée servira d'horloge pour ces bascules ?
 - ☐ X ☐ Y ☐ A ☐ C
 - Quel devra être le front actif pour ces bascules ?
 - ☐ front montant ☐ front descendant

1.2 Logique séquentielle : génération de séquence

Soit le montage ci-contre. Recopiez et complétez le tableau en donnant l'état des sorties après chaque top d'horloge (jusqu'à rebouclage de la séquence), sachant qu'à l'instant t_0 les sorties des bascules sont toutes à 0.

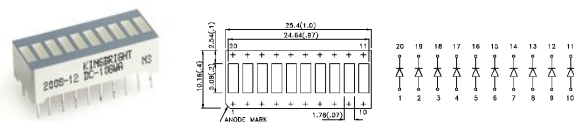
Clk	Q2	Q1	Q0
	0	0	0
↑			
↑			
... etc ...			



Symbole	Entrées		Sortie	Remarques
	J	K	Q_{n+1}	
	0	0	Q_n	Aucun changement
	0	1	0	Mise à zéro de la sortie
	1	0	1	Mise à un de la sortie
	1	1	$\overline{Q_n}$	Complément de la sortie

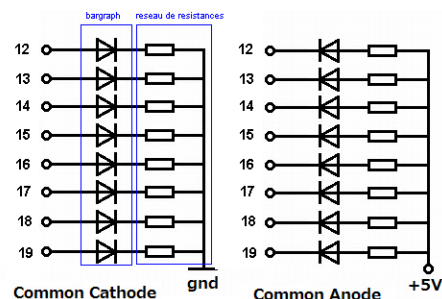
2 Pilotage d'un afficheur type Bargraph

Les afficheurs de type bargraph sont souvent utilisés. Il s'agit d'une échelle constitué d'une dizaine de LEDs. La difficulté de mise en oeuvre de ces afficheurs provient du grand nombre de lignes de commande nécessaires pour les piloter, incompatible avec des petits microcontrôleurs *low pin count*.



2.1 Liaison directe, en parallèle

On relie le bargraph (8 LEDs) au GPIOs 12,13,14,15,16,17,18,19, comme on l'a fait pour l'afficheur 7segments du TP3. Le Bargraph ne disposant pas d'anode ou de cathode commune câblé en interne, vous pouvez décider de la façon de brancher celui-ci. Dans un premier temps, on relie le bargraph selon un montage cathode commune (montage de gauche).



- Pour un GPIO X, quelle instruction allume la led X :

☐ digitalWrite(X, LOW) ☐ digitalWrite(X, HIGH) ☐ digitalRead(X)

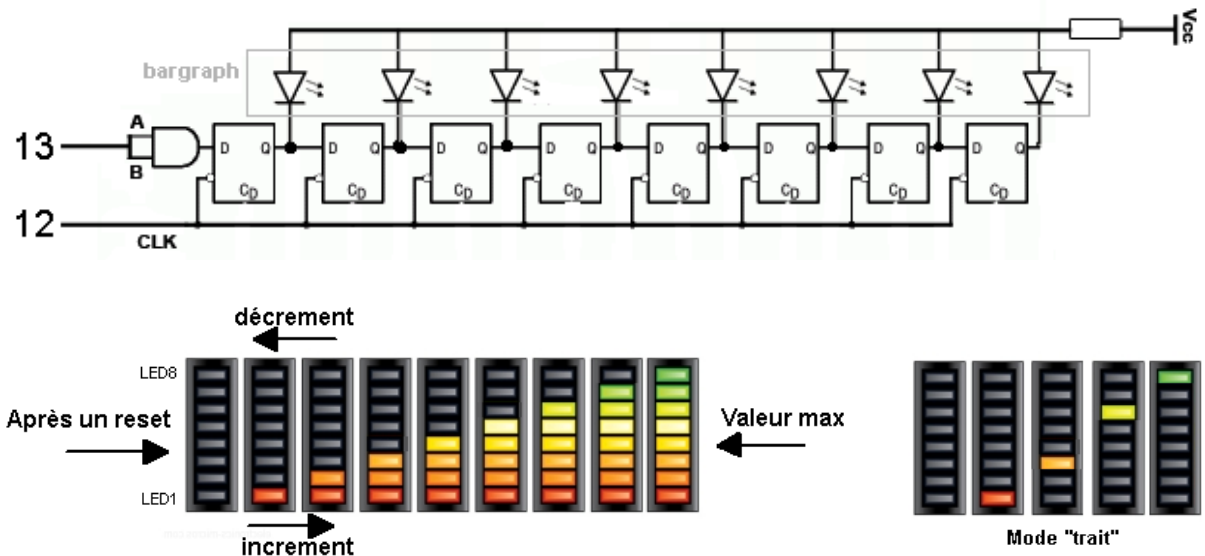
- On souhaite réaliser une animation comme celle illustrée par le gif animé (visible sur moodle, section TD), avec seulement 8 LEDs (le montage représenté dans le GIF en comporte 10), donc une LED allumée qui se déplace de la gauche vers la droite, puis de la droite vers la gauche, etc... On veut utiliser l'accès direct aux registres (comme utilisé au TP3 avec l'afficheur 7 segments) et les opérateurs de décalage. Écrire le code (uniquement la partie loop(), en utilisant delay(), pas de mise en veille. On suppose que les GPIOs 12 à 19 sont initialisés comme OUTPUT).

2.2 Utilisation d'un registre à décalage, bus série à deux fils (préparation TP8)

On décide de n'utiliser que deux GPIOs pour piloter ce bargraph, à la façon d'un bus série minimaliste. Pour cela, on ajoute un composant externe : un registre à décalage 74HC164 sur 8 bits branché comme illustré ci-dessous. Les 8 bascules D ainsi que la porte AND à gauche font partie intégrante du circuit 74HC164, voir table de fonctionnement à la fin du document.

On utilise un ESP32 et on relie l'horloge CLK du registre 74HC164 au GPIO 12, et l'entrée série (A et B du 74HC164 reliés ensemble) au GPIO 13. On suppose ces deux GPIOs comme initialisés en OUTPUT. La broche CLR du registre à décalage est relié à 1 (ne figure pas sur le schéma ci-après, mais il faut le faire en pratique).

Bien sûr, avec seulement deux GPIOs à commander, l'accès direct au registre ne s'impose plus !


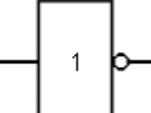
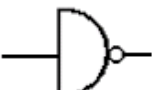
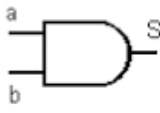
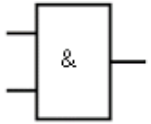
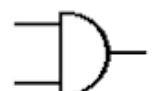
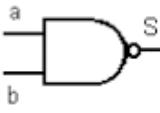
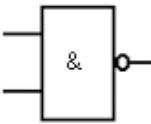
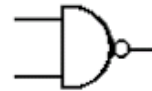
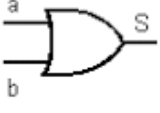
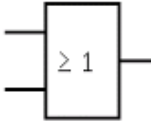

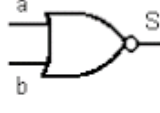
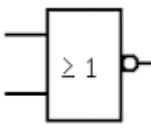

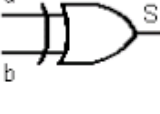
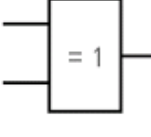
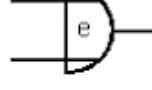
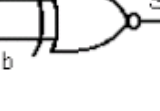
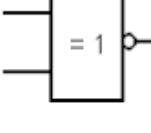
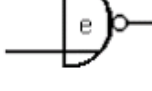


Vous remarquerez que le bargraph est maintenant connecté au registre à décalage en anode commune (c'est assez classique, les composants utilisés en électronique numérique acceptent mieux ce type de branchement). Ici, une seule résistance de limitation de courant est utilisée, dans la pratique l'usage d'un réseau de résistances est préférable.

Pour le code demandé ci-après, penser à écrire des fonctions et à réutiliser ces fonctions....

1. Écrire une fonction `BargraphClear()`. Il vous faut réfléchir aux points suivants : Quel sera l'état du bargraph après une mise sous tension ? Comment puis-je éteindre toutes les LEDs ? Le registre à décalage réagit sur quelle transition (LOW → HIGH ou HIGH → LOW)
2. Écrire une fonction `SetValue(unsigned char value)` qui affiche sur le bargraph la valeur passée en paramètre. La valeur de `value` peut varier de 0 à 8.
3. L'IDE Arduino utilise une version simplifiée du C++, on peut donc développer une classe. Écrire une classe `Bargraph`, ayant comme données membres la valeur maximale `_maxval` (renseignée lors de l'appel au constructeur) et la valeur affichée `_val`. Le constructeur aura comme paramètre le numéro des GPIOs utilisés pour CLK et l'entrée AB du 74HC164, et qui seront également mémorisés en tant que données membres.
On définira une méthode `Clear()`, une méthode `SetValue(unsigned char value)` mais aussi une méthode `Inc()` et une méthode `Dec()` qui respectivement incrémente et décrémente la donnée membre `_val` et réactualise l'affichage sur le bargraph.
4. Vous remarquerez que, pour le schéma utilisant le registre à décalage et le bargraph, on utilise qu'une seule résistance de limitation pour l'ensemble des LEDs du bargraph. Quel sera l'impact visuel de l'utilisation d'une résistance de limitation unique selon que l'on utilise un mode de visualisation "affichage colonne" ou "affichage trait" (voir illustration ci-dessus, à droite) ?
5. Modifiez votre classe pour avoir une donnée membre supplémentaire, mémorisant le mode de fonctionnement (colonne ou trait). Une méthode `Mode(int mode)` doit permettre de passer d'un mode d'affichage à l'autre.

Symboles des fonctions logiques combinatoires

FONCTION	EQUATION	SYMBOLES			TABLES DE VERITE	
		International	Français	Allemand		
NON	$S = \bar{a}$				a	S
					0	1
					1	0
ET	$S = a \cdot b$				a	b
					0	0
					0	1
					1	0
					1	1
NAND	$S = \overline{a \cdot b}$				a	b
					0	0
					0	1
					1	0
					1	1
OU	$S = a + b$				a	b
					0	0
					0	1
					1	0
					1	1
NOR	$S = \overline{a + b}$				a	b
					0	0
					0	1
					1	0
					1	1
OU Exclusif	$S = a \oplus b$				a	b
					0	0
					0	1
					1	0
					1	1
NOR Exclusif	$S = \overline{a \oplus b}$				a	b
					0	0
					0	1
					1	0
					1	1

Brochage et fonctionnement du registre à décalage 74LS164

FUNCTIONAL DESCRIPTION

The LS164 is an edge-triggered 8-bit shift register with serial data entry and an output from each of the eight stages. Data is entered serially through one of two inputs (A or B); either of these inputs can be used as an active HIGH Enable for data entry through the other input. An unused input must be tied HIGH, or both inputs connected together.

Each LOW-to-HIGH transition on the Clock (CP) input shifts data one place to the right and enters into Q₀ the logical AND of the two data inputs (A•B) that existed before the rising clock edge. A LOW level on the Master Reset (MR) input overrides all other inputs and clears the register asynchronously, forcing all Q outputs LOW.

MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	MR	A	B	Q ₀	Q ₁ –Q ₇
Reset (Clear)	L	X	X	L	L – L
Shift	H	L	L	L	q ₀ – q ₆
	H	L	h	L	q ₀ – q ₆
	H	h	L	L	q ₀ – q ₆
	H	h	h	H	q ₀ – q ₆

L (l) = LOW Voltage Levels

H (h) = HIGH Voltage Levels

X = Don't Care

q_n = Lower case letters indicate the state of the referenced input or output one set-up time prior to the LOW to HIGH clock transition.

