LE DIMNA Mathieu
mathieu.le-dimna@yncrea.fr

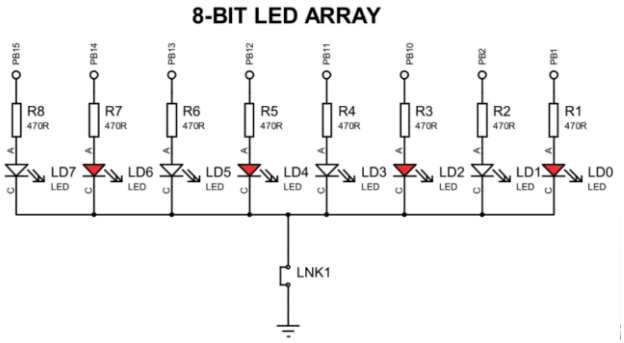
Chenillard

Objectif

Un chenillard est une animation qui allume tour à tour une ligne de LED. Cet exercice utilise les notions de structures, de tableaux et de boucles for. L'animation pilote les LED de cette manière :

[LD7, LD6, LD5, LD4, LD3, LD2, LD1, LD0]

1. [0, 0, 0, 0, 0, 0, 0, 1]
2. [0, 0, 0, 0, 0, 0, 1, 0]
3. [0, 0, 0, 0, 0, 1, 0, 0]
4. [0, 0, 0, 0, 1, 0, 0, 0]
5. [0, 0, 0, 1, 0, 0, 0, 0]
6. [0, 0, 1, 0, 0, 0, 0, 0]
7. [0, 1, 0, 0, 0, 0, 0, 0]
8. [1, 0, 0, 0, 0, 0, 0, 0]



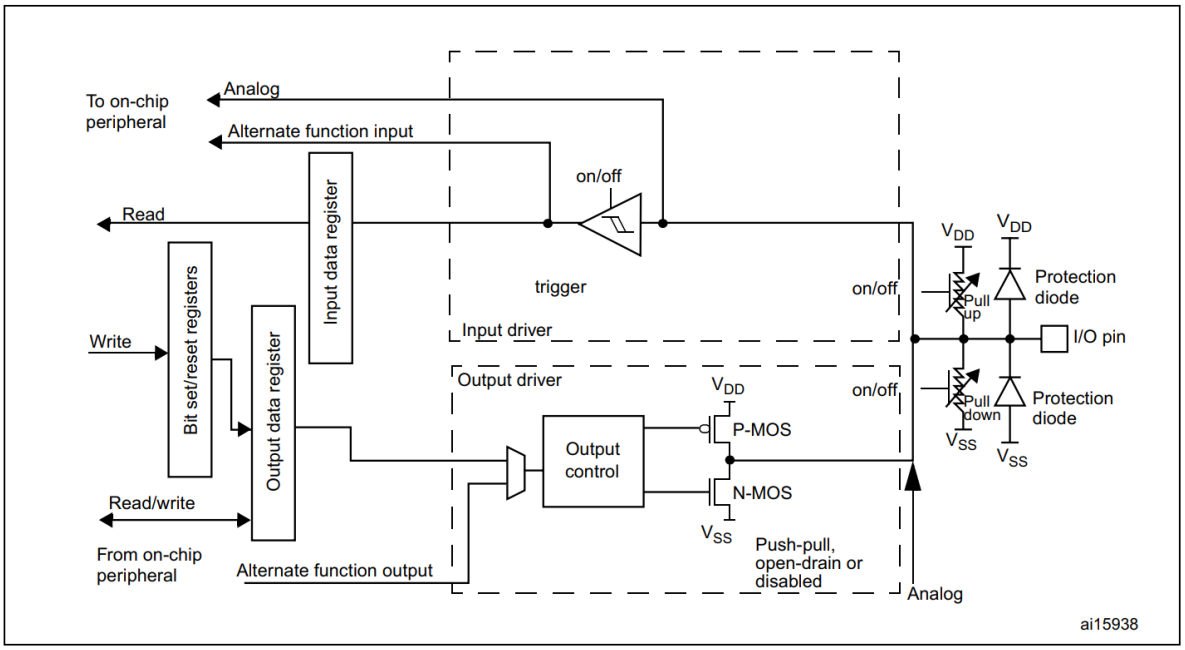
Créer le projet Chenillard

Créez un nouveau projet nommé *Chenillard* en accédant à ***File > New > STM32 Project***. La cible est le *STM32L152RET6*, choisissez ce MCU dans le menu ***MCU/MPU Selector***.

Fichier IOC

Paramétrez les GPIO associés aux LED en sortie de type Push-Pull[1] sans pull-up ni pull-down.

Figure 18. Basic structure of a standard I/O port bit



*Note : Il est possible d'ajouter un label aux entrées/sorties depuis la vue **Pinout & Configuration > System Core > GPIO** et modifier le champ **User Label** associé au GPIO dans la liste. Vous pourrez les nommer respectivement LD7, LD6, LD5, LD4, LD3, LD2, LD1, LD0.*

[1] NMOS et PMOS complémentaires qui forcent à VDD (3V3) ou à VSS (0V) les pins du STM32 associées

LE DIMNA Mathieu
mathieu.le-dimna@yncrea.fr

Code utilisateur

Dans le fichier ***Core > Inc > main.h*** vous trouverez une liste de définitions[2] représentant les LED. Pour créer un tableau qui stockera ces informations pour le chenillard créez une nouvelle définition qui représentera la taille de ce tableau. *Attention, il y a un espace dédié aux définitions de l'utilisateur. Tout changement effectué hors espace dédié pourra être écrasé lors d'une nouvelle génération du fichier IOC.*

Une fois les modifications effectuées ouvrez le fichier ***Core > Src > main.c***. Pour automatiser le plus simplement possible le chenillard il faudra préparer quelques outils :

- La définition d'une structure qui contiendra les informations d'une LED (Port GPIO et Pin). *Le fichier main.c inclut le header **main.h**. Les définitions contenues dans ce fichier sont donc accessibles par votre code dans ce fichier.*
- La définition d'un tableau qui contiendra l'ensemble des LED définies comme la structure que vous avez créé avant. Vous pourrez changer le contenu des structures du tableau en modifiant les champs manuellement dans votre ***main***. *La taille du tableau est définie dans le fichier main.h.*

Une fois ces variables préparées vous pourrez placer votre algorithme dans le contexte de la boucle ***while (1)***. *Je rappelle, il y a un espace dédié au code utilisateur. Tout changement effectué hors espace dédié pourra être écrasé lors d'une nouvelle génération du fichier IOC.*

Notez que vos LED sont définies dans un tableau maintenant, vous pourrez accéder à chacune d'entre elles avec une variable incrémentée par une boucle **for**.

Pour forcer les GPIO à 1 ou 0 vous utiliserez les fonctions HAL associées à ce périphérique. Dans ***Drivers > STM32L1xx_HAL_Driver > Inc*** trouvez le fichier .h associé aux GPIO et cherchez une fonction dédiée à l'écriture sur une pin GPIO. Aussi, pour observer les changements de LED, vous devrez attendre un certain temps sinon l'animation s'exécutera trop vite… Recherchez une fonction de délai dans les fichiers HAL.

Pour aller plus loin

Le problème avec les fonctions de délai c'est qu'elles sont bloquantes : pendant ce temps on ne fait rien et le processeur est sollicité. Trouvez un moyen d'exécuter le changement de LED toutes les 500ms sans utiliser la fonction HAL_Delay.

[2] #define : Définit un élément que le préprocesseur remplacera par sa valeur associée.